Electronic Colloquium on Computational Complexity, Report No. 134 (2016)

Explicit List-Decodable Codes with Optimal Rate for Computationally Bounded Channels

Ronen Shaltiel and Jad Silbak

August 29, 2016

Abstract

A stochastic code is a pair of encoding and decoding procedures $(\mathrm{Enc}, \mathrm{Dec})$ where $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$, and a message $m \in \{0,1\}^k$ is encoded by $\mathrm{Enc}(m,S)$ where $S \leftarrow \{0,1\}^d$ is chosen uniformly by the encoder. The code is (p,L)-list-decodable against a class $\mathcal{C}$ of "channel functions" $C : \{0,1\}^n \to \{0,1\}^n$ if for every message $m \in \{0,1\}^k$, and every channel $C \in \mathcal{C}$ that induces at most pn errors, applying Dec on the "received word" $C(\mathrm{Enc}(m,S))$ produces a list of at most L messages, that contains m with high probability (here the probability is over the choice of $S \leftarrow \{0,1\}^d$). Note that both the channel C, and the decoding algorithm Dec, *do not* receive the random variable S. The rate of a code is $R = k/n$, and a code is explicit if $\mathrm{Enc}, \mathrm{Dec}$ run in time $\mathrm{poly}(n)$.

Guruswami and Smith (Journal of the ACM, to appear), showed that for every constants $0 < p < \frac{1}{2}$ and $c > 1$ there are *Monte-Carlo* explicit constructions of stochastic codes with rate $R \geq 1 - H(p) - \epsilon$ that are $(p, L = \mathrm{poly}(1/\epsilon))$-list decodable for size n^c channels. Here, Monte-Carlo, means that the encoding and decoding need to share a public uniformly chosen $\mathrm{poly}(n^c)$ bit string Y, and the constructed stochastic code is (p,L)-list decodable with high probability over the choice of Y.

Guruswami and Smith pose an open problem to give fully explicit (that is not Monte-Carlo) explicit codes with the same parameters, under hardness assumptions. In this paper we resolve this open problem, using a minimal assumption: the existence of poly-time computable pseudorandom generators for small circuits, which follows from standard complexity assumptions by Impagliazzo and Wigderson (STOC 97).

Guruswami and Smith also asked to give a fully explicit unconditional constructions with the same parameters against $O(\log n)$-space online channels. (These are channels that have space $O(\log n)$ and are allowed to read the input codeword in one pass). We also resolve this open problem.

Finally, we consider a tighter notion of explicitness, in which the running time of encoding and list-decoding algorithms does not increase, when increasing the complexity of the channel. We give explicit constructions (with rate approaching $1 - H(p)$ for every $p \leq p_0$ for some $p_0 > 0$) for channels that are circuits of size $2^{n^{\Omega(1/d)}}$ and depth d. Here, the running time of encoding and decoding is a fixed polynomial (that does not depend on d).

Our approach builds on the machinery developed by Guruswami and Smith, replacing some probabilistic arguments with explicit constructions. We also present a simplified and general approach that makes the reductions in the proof more efficient, so that we can handle weak classes of channels.

ISSN 1433-8092

1 Introduction

List decodable codes. List decodable codes are extensively studied in Coding Theory and Theory of Computer Science, and have many applications. In the paragraph below we define list-decodable codes, using a functional view, which is more convenient for this paper.

A code is defined by a pair $(\mathrm{Enc}, \mathrm{Dec})$) of encoding and decoding procedures. We say that $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$, is (p, L)-list decodable, if there exits a function Dec which given $y \in \{0,1\}^n$, $\mathrm{Dec}(y)$ produces a list of size L containing all elements $m \in \{0,1\}^k$ such that $\delta(y, \mathrm{Enc}(m)) \leq p$, (here $\delta(x, y)$ is the relative hamming distance of x and y). Unique decoding is the special case where $L = 1$, and a code is explicit if both encoding and decoding can be performed in time polynomial in n. The rate of a code is $R = \frac{k}{n}$. (A more detailed formal definition is given in Section 3.2).

Towards explicit capacity-achieving, binary list decodable codes. It is known that for $0 < p < \frac{1}{2}$, binary (p, L)-list decodable codes must have rate $R \leq 1 - H(p)$ for nontrivial size lists, and a longstanding open problem in coding theory is to give an explicit construction of binary codes matching list-decoding capacity. That is, show that for every constants $0 < p < \frac{1}{2}$, and $\epsilon > 0$, and for every sufficiently large n, there are explicit binary list decodable codes with rate $R = 1 - H(p) - \epsilon$, that are (p, L)-list decodable, for a constant L that depends on ϵ. The probabilistic method shows that there exist nonexplicit codes with these parameters. (In fact, the probabilistic method achieves list size L which is $\mathrm{poly}(1/\epsilon)$, and this is a benchmark that can be compared to.) Today, despite substantial effort, no explicit constructions are known, even if we insist only on explicit encoding, and do not require list-decoding to be explicit.

Restricted channels. Explicit uniquely decodable, binary codes achieving rate approaching $1 - H(p)$, are known for restricted classes of channels. There is a large body of work in Shannon's framework, on channels which are not *adversarial* and inflict "random errors". The most famous example is a binary symmetric channel, that flips each symbol independently with probability p, and there are explicit, uniquely decodable, binary codes with rate approaching $1 - H(p)$ for such channels.

Computationally bounded channels. Lipton [Lip94] considered intermediate classes of *adversarial* channels according to the computational complexity of the channel. More specifically, we can think of a channel as a function $C : \{0,1\}^n \to \{0,1\}^n$ and consider families channels that:

- Induce at most pn errors. That is, for every $z \in \{0,1\}^n$, $E_C(z) := z \oplus C(z)$ has hamming weight at most pn.
- Are computationally bounded. That is, we only consider C that belong to some complexity class $\mathcal{C}$.

Natural examples of complexity classes are polynomial size circuits and logarithmic space branching programs. Note that these two classes are nonuniform, and it is more natural to use nonuniform classes, as such classes trivially contain channels C where E_C is constant (meaning that there is a fixed error vector e such that $C(z) = z \oplus e$). Such channels are called "additive channels" and as they are the simplest form of adversarial behavior, it makes sense that we allow them in any class of computationally bounded channels.

Another advantage of using nonuniform classes of channels, is that it is sufficient to consider *deterministic* channels, in order to obtain security against *randomized* channels. This is because by averaging, if there is a computationally bounded randomized channel that is able to prevent decoding on some message m, then we can fix its random coins and obtain a deterministic channel (which is hardwired with a good choice of random coins).

1.1 Stochastic codes

Unfortunately, the notion of computationally bounded channels is not interesting in the standard setup of error correcting codes: It is easy to show that if a code $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ is list-decodable against additive channels, then it is list-decodable against unbounded channels.[1]

Several setup assumptions were introduced in order to circumvent this problem. In this paper, we are interested in a setup of "stochastic codes" studied by Guruswami and Smith [GS10]. We remark that other setups have been considered and we mention these in Section 1.4.

Let $\mathcal{C}$ be a class of channels that induce at most pn errors. A stochastic code against $\mathcal{C}$ consists of a pair of algorithm $(\mathrm{Enc}, \mathrm{Dec})$ such that:

- The encoding algorithm $\mathrm{Enc}(m, S)$ receives a message $m \in \{0,1\}^{Rn}$ and a uniform string S (that is not known to the channel or decoding algorithm) and outputs an n bit string that is the codeword.
- A channel $C \in \mathcal{C}$, that **does not** receive the string S, corrupts the codeword, generating $C(\mathrm{Enc}(m, S))$.
- The decoding algorithm gets the "corrupted codeword" $C(\mathrm{Enc}(m, S))$, but **does not** receive the string S.
- For every message m, and for every channel $C \in \mathcal{C}$, the decoding done by $\mathrm{Dec}(C(\mathrm{Enc}(m, S)))$ needs to successfully recover the original message m with probability $1 - \nu$ over the choice of S. ($\nu > 0$ is an error parameter).

  Here, "success" means to output m (in case of unique decoding) or output a list of size L that contains m (in case of list decoding).

A formal definition follows:

Definition 1.1 (Stochastic Codes). *Let k, n, q be parameters and let $Enc : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ 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** with success probability $1-\nu$ against channels in $\mathcal{C}$, if there exists a function $Dec : \{0,1\}^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $C \in \mathcal{C}$, $\Pr_{S \leftarrow U_d}[Dec(C(Enc(m, S))) = m] \geq 1 - \nu$. We typically, parameterize $\mathcal{C}$ with two parameters: the complexity of functions in the class, and the number errors that they induce.*

[1] Specifically, if a code is not (combinatorially) list-decodable, then there exist a received word y that has too many codewords that are close to it. Let c be one of these codewords, and let $e = c \oplus y$ and consider the additive channel $C_e(z) = z \oplus e$. This channel "breaks" the code as $C(c) = c \oplus e = y$, and y is a received word on which decoding cannot succeed.

- *L-**list-decodable** with success probability $1-\nu$ against channels in $\mathcal{C}$ if the function Dec above is allowed to output a list of size at most L that contains m.*

*A code is **explicit** if its encoding and decoding functions are computable in time polynomial in their input and output. The rate of the code is the ratio of the message length and output length of Enc.*

Guruswami and Smith [GS10] gave explicit constructions of stochastic codes with rate approaching $1 - H(p)$ (for $0 < p < \frac{1}{2}$) that are uniquely decodable against additive channels. They also showed that for $p > 1/4$ there are computationally weak channel families, against which, stochastic codes with rate approaching $1-H(p)$ and unique decoding do not exist. (All the complexity classes considered in this paper can simulate these weak channels.)

A Monte-Carlo construction of stochastic codes for poly-size circuits. Guruswami and Smith [GS10] showed that for every constant c, there is a Monte-Carlo explicit construction of list decodable stochastic codes against channels of size n^c, with rate approaching $1 - H(p)$. By Monte-Carlo, we mean that:

- The encoding and decoding algorithms receive an additional input y of length $\text{poly}(n^c)$.
- With high probability over the choice of y, the encoding and decoding algorithms (that are hardwired with y) form the required stochastic code.[2]

1.2 Our results

Guruswami and Smith stated the following open problem: give fully explicit (that is *not* Monte-Carlo) constructions of stochastic codes against poly-size circuits, under complexity theoretic assumptions.

Necessity of complexity theoretic assumptions. As we explain later, complexity theoretic assumptions are not necessary in order to give *Monte-Carlo* constructions of stochastic codes. They are necessary to give fully explicit constructions (which are not Monte-Carlo) in the following sense: Given a stochastic code against circuits of size n^c, we can consider the "optimal channel" that given a codeword $z \in \{0,1\}^n$, tries all possible error vectors $e \in \{0,1\}^n$ of relative hamming weight p, and finds the first one on which decoding fails, if such a vector exist. This channel succeeds iff the code isn't secure against unbounded channels. If the code isn't secure against unbounded channels (but secure against size n^c channels) then this attack cannot be carried out in size n^c. This means that there is a problem computable in $\text{E} = DTIME(2^{O(n)})$ that for every sufficiently large n, cannot be solved by size n^c circuits.[3] We remark that this type of assumptions (namely, that there is a problem in E that requires large circuits) is exactly the type of assumption that implies and is implied by, existence of explicit pseudorandom generators in the Nisan-Wigderson setting [NW94, IW97].

[2] We mention that the approach of Guruswami and Smith dictates that the length of y is larger than n^c (and in general larger than the log of the number of allowed channels). This means that a channel is not "sufficiently complex" to receive y as input.

[3] In fact, for this argument, we don't need the stochastic code to be explicit. Encoding is allowed to be arbitrary, and decoding is allowed to run in time $2^{O(n)}$.

1.2.1 Explicit stochastic codes for poly-size circuits

Our first result resolves the open problem posed by Guruswami and Smith, and we construct explicit stochastic codes against poly-size circuit channels, under an assumption that is only slightly stronger than what is implied by the existence of such codes.

Theorem 1.2 (Explicit stochastic codes for poly-size channels). *If there exists a constant $\beta > 0$ and a problem in $E = DTIME(2^{O(n)})$ such that for every sufficiently large n, solving E on inputs of length n, requires circuits of size $2^{\beta \cdot n}$, then for every constants $0 < p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for size n^c circuits that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant.*

Theorem 1.2 is stated in more detailed form in Theorem 5.4. The assumption used in the Theorem is a standard complexity assumption, and was used by Impagliazzo and Wigderson [IW97] to show that BPP=P.

1.2.2 Unconditional explicit stochastic codes for space $O(\log n)$ online channels

Guruswami and Smith also considered "space s online channels". These are channels $C : \{0,1\}^n \to \{0,1\}^n$ implemented by space s (or equivalently width 2^s) oblivious read-once branching programs (ROBPs). Below is a standard definition of ROBPs tailored for functions that output many bits.

Read Once Branching Programs. We will only be interested in space $s \geq \log n$. A space s ROBP $C : \{0,1\}^n \to \{0,1\}^n$ is defined using a layered graph with $n+1$ layers, where the first layer has a single node v_0, and remaining layers have 2^s nodes. Each node v in the first n layers has two outgoing edges (labeled with zero and one) connected to nodes in the next layer, and each node v is also labeled by an "output bit" $b(v)$. On input $x \in \{0,1\}^n$, the computation of C is defined by following the unique path from v_0 to the last layer, defined by taking the edge marked with x_i at step i. The output $C(x)$ is the concatenation of the n output bits, collected at nodes along the path. It is standard that for $s \geq \Omega(\log n)$ ROBPs with space $O(s)$ capture the nonuniform version of space $O(s)$ computation, that reads its n bit input x in fixed order. We remark that all the results in this paper also hold if we allow channels to have s bits of "lookahead", allowing them to also read the bits $i+1, \ldots, i+s$ before outputting the i'th bit.

Guruswami and Smith stated the following open problem: give unconditional fully explicit (that is *not* Monte-Carlo) constructions of stochastic codes against space $O(\log n)$ online channels.[4] Our second result resolves this open problem.

Theorem 1.3 (Explicit stochastic codes for space $O(\log n)$ online channels). *For every constants $0 < p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for space $c \log n$ online channels that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant.*

Theorem 1.3 is stated in more detailed form in Theorem 5.5.

[4] A preliminary version of [GS10] contained an unconditional Monte-Carlo construction of stochastic code against space $O(\log n)$ online channels, and a conditional Monte-Carlo construction for size n^c circuits (relying on the existence of "Nisan-Wigderson style", pseudorandom generators for size n^c circuits). However, Monte-Carlo constructions can easily obtain "Nisan-Wigderson style" pseudorandom generators, as a random function with polynomial size description is w.h.p. such a generator. Consequently, no hardness assumption is needed for Monte-Carlo constructions against polynomial size circuits, which are secure also against $O(\log n)$ space online channels.

Efficiency of encoding/decoding versus channel complexity. The approach of Guruswami and Smith [GS10] (that we also use) dictates that security can only be proven for channel families that are not sufficiently strong to run the decoding algorithm.[5] Consequently, in the Monte-Carlo construction and our Theorem 1.2, the running time of encoding and decoding is a polynomial in n that is larger than the circuit size n^c. It is an intriguing open problem whether stochastic codes with rate approaching $1 - H(p)$, that can be encoded and decoded in fixed polynomial time (say n^3) against any polynomial size channel, can be constructed (under cryptographic assumptions). We do not know whether this is possible.

We can however expect to obtain fixed polynomial time (that does not depend on the constant c) for encoding and decoding in our Theorem 1.3. Unfortunately, this is not the case, and the encoding and decoding algorithm that we obtain in Theorem 1.3 run in time polynomial in n^c (and in particular larger than n^c) when working against space $c \log n$ channels. We do not know how to avoid this dependence.

1.2.3 Stochastic codes for AC^0 channels, with fixed poly-time encoding/decoding

We are able to obtain fixed polynomial time algorithms for encoding and decoding for a family of channels implemented by superpolynomial size and constant depth circuits. For technical reasons, we achieve this only for $p \leq p_0$ for some $p_0 > 0$. The result is stated below.

Theorem 1.4 (Explicit stochastic codes for AC^0 channels). *There exist constants $p_0 > 0$ and $a > 0$ such that for every constants $0 < p \leq p_0$, $\epsilon > 0$, d, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for size $2^{n^{\frac{1}{ad}}}$ circuits of depth d that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant. (Here, encoding and decoding run in fixed polynomial time that does not depend on d, and only the choice of which n is sufficiently large, depends on d.)*

The constant p_0 comes from a specific construction of AG-codes, and it seems that p_0 can be pushed to be any constant strictly smaller than $1/12$. Theorem 1.4 is stated in more detailed form in Theorem 5.6.

1.3 Perspective

Explicit codes against computationally bounded channels give the "best of both worlds": They can recover from errors induced by *adversarial* channels, while having information theoretic optimal rate approaching $1 - H(p)$.

As pointed out by Guruswami and Smith, essentially all randomized channels studied in the Shannon framework of error correcting codes, are computationally simple (and it seems that all of them can be implemented by constant depth circuits or online logspace). This means that the computational perspective leads to a unified construction of explicit codes that are good for all "Shannon style" randomized channels simultaneously, while also being able to recover against many adversarial channels (and in particular against additive channels).

We believe that the distinction we make above (namely, whether encoding/decoding efficiency is allowed to increase with the complexity of the channel) is important so that the added benefit of

[5] The approach of Guruswami and Smith (that we also use) relies on the fact that channels cannot distinguish between encodings of two messages. Therefore, if decoders aren't stronger than channels, they cannot hope to decode, even if there are no errors.

codes for computationally bounded channels doesn't come with a price tag of being less efficient. Specifically, our construction for AC^0 channels uses "regular" coding theoretic ingredients and does not have to "pay extra" for being able to handle channels that are superpolynomial size circuits of constant depth.

An intriguing open problem is whether unique decoding is possible for computationally bounded channels with rate approaching $1 - H(p)$. Guruswami and Smith [GS10] showed that this is impossible for $p > 1/4$ (and their argument works for all classes of channels discussed in this paper). It is not known whether unique decoding is possible for $p < 1/4$ for the channel classes that we consider.

1.4 Some related work

The notion of computationally bounded channels was initially studied in cryptographic 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 view it as a version of stochastic codes in which the decoding algorithm **does** receive the S.

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, 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.

We also point out that the Monte-Carlo construction of Guruswami and Smith, also requires less setup. While the encoder and decoder do need to share a random string, this string does not need to be private. It can be chosen once and revealed to the channel.

Private Codes. A related notion of "private codes" was studied by Langberg [Lan04]. Here channels are unbounded, codes are existential (and not explicit), and 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.

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).

2 Overview of the technique

In this section we give a high level overview of the construction. Our construction heavily relies on previous work in the area (mainly on that of Guruswami and Smith [GS10]). In this high level overview we attempt to highlight our technical contribution, while also giving a high level overview of the many ideas from previous work that are used in the construction. Therefore, we start with a high level description of earlier work, and build up to the work of Guruswami and Smith. Along the way, in Section 2.2 we explain the modifications that allow us to handle weak classes of channels. Finally, in Section 2.4, we present a self contained problem (that of constructing inner stochastic codes). Constructing such explicit codes is the main source of our improvement over Guruswami and Smith, and we give a high level overview of our approach.
The reader can skip this high level overview and go directly to the technical section.

2.1 Codes for the setup of shared private randomness

We start by explaining how to construct codes with rate approaching $1-H(p)$ in the case that the setup allows shared private randomness. Recall that this can be thought of as a stochastic code in which the decoding algorithm receives the random string chosen by the encoding. We present the ideas that are used to construct codes against bounded channels in this setup, in two steps. We first explain how to handle additive channels, and then explain how this method can be extended to handle bounded channels that are not additive. The ideas from both these reductions are key components in the construction of Guruswami and Smith.

Reducing additive channels to binary symmetric channels. We start by constructing codes with shared private randomness against additive channels. The encoder and decoder will share a description S_π of a uniformly selected permutation $\pi : [n] \to [n]$. The encoding will be defined by

$$\mathrm{Enc}(m, S_\pi) = \pi(\mathrm{Enc}_{BSC}(m)),$$

meaning that Enc encodes m by a code for binary symmetric channels, and then uses the permutation π to rearrange the n indices of the encoding, placing the i'th bit, in the $\pi(i)$'th position. Note that for any additive channel $C_e(z) = z \oplus e$ that induces pn errors, the effect of the channel on $\mathrm{Enc}(m, S_\pi)$ can be essentially viewed as applying a binary symmetric channel on $\mathrm{Enc}_{BSC}(m)$, meaning that the decoder is able to uniquely decode against additive channels, with a code that has rate approaching $R = 1 - H(p)$ (which can be achieved explicitly for binary symmetric channels).

Smith [Smi07] showed that an (almost) t-wise independent permutation can be coupled with specific constructions of codes for binary symmetric channels, and used instead of a truly random permutation. This reduces the length of the shared key and allows keys shorter than n.

Reducing computationally bounded channels to additive channels. It is possible to use cryptography (or more generally pseudorandomness) to handle computationally bounded channels: Assume that in addition to the seed S_π, the encoder and decoder also share a seed S_{PRG} for a pseudorandom generator PRG that fools computationally bounded channels and outputs n bits, and define:

$$\mathrm{Enc}'(m, (S_\pi, S_{PRG})) = \mathrm{Enc}(m, S_\pi) \oplus PRG(S_{PRG}) = \pi(\mathrm{Enc}_{BSC}(m)) \oplus PRG(S_{PRG}).$$

This means that the rate of Enc' is inherited from Enc and can approach $1 - H(p)$. A useful property is that for every fixed s_π, the random variable $\mathrm{Enc}(m, (s_\pi, S_{PRG}))$ is pseudorandom for the channel. This can be used to show that a computationally bounded channel cannot prevent correct decoding.

We now explain this argument. The decoding algorithm $\mathrm{Dec}'(y, (s_\pi, s_{PRG}))$ will simply compute $y' = y \oplus PRG(s_{PRG})$ and apply the previous decoding algorithm Dec on y' and s_π. We show that for every computationally bounded channel C that induces at most pn errors, the decoding succeeds with probability at least $1 - (\nu + \epsilon_{PRG})$, where ϵ_{PRG} is the error of the generator PRG.

We consider the function $A(m, s_\pi, e)$ that checks if $\mathrm{Dec}_{BSC}(\mathrm{Enc}(m, s_\pi) \oplus e)$ successfully recovers m. In the previous section we've seen that for every message m, and error vector e of relative hamming weight at most p, $\Pr[A(m, S_\pi, e)] \geq 1 - \nu$. Consequently, for every channel C that induces pn errors,

$$\Pr[A(m, S_\pi, E_C(U_n)) = 1] \geq 1 - \nu$$

(this follows as U_n is independent of S_π, and recall that $E_C(z) = z \oplus C(z)$). If decoding does not work, and there exist a message m such that:

$$\Pr[A(m, S_\pi, E_C(\mathrm{Enc}'(m, (S_\pi, S_{PRG})))) = 1] < 1 - (\nu + \epsilon_{PRG}).$$

By averaging over S_π, this gives that there exists a fixed value s_π such that:

$$\Pr[A(m, s_\pi, E_C(U_n)) = 1] - \Pr[A(m, s_\pi, E_C(\mathrm{Enc}'(m, (s_\pi, S_{PRG})))) = 1] > \epsilon_{PRG},$$

meaning that $f(z) = A(m, s_\pi, E_C(z))$ distinguishes $\mathrm{Enc}(m, (s_\pi, S_{PRG}))$ from U_n with probability ϵ_{PRG}, which is a contradiction if PRG is ϵ_{PRG}-pseudorandom against f (which is essentially the composition of the channel and Dec_{BSC}). As Dec_{BSC} runs in polynomial time, it follows that a PRG against poly-size circuits suffices to handle poly-size channels.

2.2 A more efficient reduction for online logspace and AC^0.

A drawback of the approach described above is that while the decoding algorithm Dec_{BSC} runs in polynomial time, existing constructions rely on decoding an "outer code" (typically, Reed-Solomon) which cannot be done by small constant depth circuits or small space ROBPs. In this paper we are interested in channels that run in online logspace or AC^0. We would like to use PRGs that fool these classes (and explicit constructions are unconditional) rather than PRGs for poly-size circuits (which are inherently conditional as they imply circuit lower bounds).

For this purpose we replace the code $(\mathrm{Enc}_{BSC}, \mathrm{Dec}_{BSC})$ (for binary symmetric channels) by a code $(\mathrm{Enc}_{\mathrm{balanced}}, \mathrm{Dec}_{\mathrm{balanced}})$ that is list decodable from *balanced errors*. We now define this notion. A string $e \in \{0,1\}^n$ is (b', p, γ)-balanced if when viewed as $e \in (\{0,1\}^{b'})^{n/b'}$, at most a γ fraction of blocks of e, have relative hamming weight larger than p. It is not hard to construct explicit codes which are list-decodable (with constant size lists) against error vectors that are (b', p, γ)-balanced and have rate approaching $1 - H(p)$ for small constant γ. We give such a construction in Section 3.2.1.

If we take an error vector of Hamming weight p and permute it using a random (or t-wise independent) permutation, then with high probability it will indeed be $(b', p + \alpha, \gamma)$-balanced for sufficiently large b', and small constant $\alpha, \gamma > 0$. This means that codes against balanced errors in particular work against binary symmetric channels.

The advantage of this notion is that the function A of the previous section can be made more efficient. Rather than having to decode Enc_{BSC}, it is sufficient to check if the error vector e is $(b', p+\alpha, \gamma)$-balanced, which can be performed by models that can count (or even only approximately count) such as small ROBPs, or AC^0. This leads to more efficient reductions that enable us to use PRGs for weaker classes.[6]

2.3 Stochastic codes for bounded channels

We start by presenting the approach of Guruswami and Smith [GS10] to take codes for shared private randomness (as presented in the previous section) and convert them into stochastic codes.

Let $(\mathrm{Enc}', \mathrm{Dec}')$ be the code for shared private randomness (presented in the previous section). We will reserve N for the block length of the stochastic code that we want to construct, and use N_{data} as the block length of Enc'. We have that the rate of Enc' can approach $1 - H(p)$ and so it is sufficient that the rate of the code $(\mathrm{Enc}, \mathrm{Dec})$ that we construct, approaches that of Enc'.

We will set $N = N_{\mathrm{data}} + N_{\mathrm{ctrl}}$ where $N_{\mathrm{ctrl}} = \epsilon \cdot N$ (for a small constant ϵ) so that the rate indeed approaches $1 - H(p)$. Loosely speaking, when a given a message m and "control information" S (which will include (S_π, S_{PRG}) as well as additional randomness) we will set $c_{\mathrm{data}} = \mathrm{Enc}'(m, (S_\pi, S_{PRG})) \in \{0,1\}^{N_{\mathrm{data}}}$ and $c_{\mathrm{ctrl}} \in \{0,1\}^{N_{\mathrm{ctrl}}}$ will be an encoding of S (that we specify later). We will then merge these two strings into a string $c = (c_{\mathrm{data}}, c_{\mathrm{ctrl}})$ of length N.

The high level intuition, is that the encoder encodes the control information S and embeds it in the encoding of m, hoping that the decoder can find it, decode it to get S, and then use the decoding algorithm Dec' (which requires S) to decode the data part.

However, there are two seemingly contradicting requirements: On the one hand, the decoder needs to find the "control information" in order to recover S. On the other, if it is easy to identify which part of the encoding encodes the "control information", then the channel can focus its errors on it, wiping it out completely.

Stochastic codes for additive channels. The first step taken by Guruswami and Smith is to ensure that an additive channel cannot wipe out the control information. For this purpose they divide the N output bits into $n = N/b$ blocks of length b (where b is a parameter to be chosen later). The encoder will use additional randomness S_{samp} to choose a random set $I = \{i_1, \ldots, \epsilon \cdot n\}$ of distinct indices in $[n]$. The string S_{samp} will be part of the "control information" S (making $S = (S_{\mathrm{samp}}, S_\pi, S_{PRG})$) and in order to make its length less than n, the sampling is made by a randomness efficient averaging sampler (see Section 3.3 for details). We will pretend that the set I is completely random in this high level presentation.

The set I will define which blocks are "control blocks", and the final embedding of $c_{\mathrm{data}}, c_{\mathrm{ctrl}}$ into an N bit string, is done by placing c_{ctrl} in the control blocks, and c_{data} in the remaining blocks (which are suitably called data blocks). The sampling of I guarantees that for every fixed error vector e of relative hamming weight at most p, at least an $\epsilon/2$ fraction of the control blocks, are not hit with significantly more than pb errors. This will suffice for the decoding algorithm.

The decoder (that does not know I) will go over all n blocks, treating each one of them as a potential control block. Even if no errors are inflicted, only $\epsilon \cdot n$ of the n blocks are indeed control

[6] Some additional effort is needed to make this idea go through for online channels, as after the permutation, input bits "arrive" in a way that doesn't respect the partitioning into blocks. A preliminary version of [GS10] contained an alternative, and more complex approach in order to deal with online channels.

blocks. We want the decoder to be able to "list-decode" and output a small list of candidates s for the "control information".

This can be done as follows: When preparing c_{ctrl}, the control information S will be encoded by a concatenated code, where the outer code is list-decodable (or more generally list-recoverable) and has block length $\epsilon \cdot n$, and the inner code has symbols of b bits, and is decodable from slightly more than pb errors. This way, if at least $\epsilon/2$ fraction of the control blocks suffer not much more than pb errors (and are therefore decoded correctly by the inner code) then the list decoding algorithm of the outer code produces a list of candidates that includes the correct control information s. Decoding can now proceed, and for each such candidate s, it can apply Dec' on the data part (defined by s_{samp}) using the control information (s_π, s_{PRG}). This indeed suffices for list decoding against additive channels.

Extending the approach to computationally bounded channels. There is an obvious concern if we use this strategy against channels that are not additive: The channel C may inspect the different n blocks, and try to identify which of them are control blocks. It is crucial that the channel will not be able to distinguish a control block from a data block. This means that we want the inner code that produces the b-bit control blocks to have three properties:

- It should be able to decode from roughly pb errors.
- The channel should not be able to distinguish control blocks from data blocks.
- Control blocks shouldn't reveal information about S to the channel.

Here, it is useful that the data part is xored with $PRG(S_{PRG})$ and is therefore pseudorandom. This means that we can obtain these three properties if we use a stochastic code (instead of a standard code) and require that the output is pseudorandom. Note that here the notion of stochastic codes is not used to "improve decoding properties" (we can do with standard codes). Instead, it is used to perform encoding in a way that does not reveal information about the message. This notion of stochastic codes is defined in the next section.

2.4 Pseudorandom stochastic inner codes

Guruswami and Smith considered the following version of stochastic codes. Let $\text{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ be a function.

1. We say that Enc is ϵ-**pseudorandom** for a class of functions $\mathcal{C}$ if for every $m \in \{0,1\}^k$ and for every $C \in \mathcal{C}$, the distribution $\text{Enc}(m, U_d)$ is ϵ-pseudorandom for C, meaning that: $|\Pr[C(\text{Enc}(m, U_d)) = 1] - \Pr[C(U_b) = 1]| \leq \epsilon$.

2. We say that Enc is L-**list decodable** with **radius** p if there exists a function Dec such that for every $y \in \{0,1\}^b$, $\text{Dec}(y)$ produces a list of at most L pairs $(m, r) \in \{0,1\}^k \times \{0,1\}^d$ that contains all pairs $(m, r) \in \{0,1\}^k \times \{0,1\}^d$ such that $\delta(y, \text{Enc}(m, r)) \leq p$.

Such codes can be plugged in as "inner control codes" in the scheme described in the previous section, and the two properties above suffice for the correctness of the construction (if pseudorandomness is guaranteed against a class sufficiently stronger than the channel, as explained in Section 2.2).

Consequently, the task of explicitly constructing stochastic codes against bounded channels reduces to explicitly constructing such stochastic codes with constant size lists. Here, we benefit from the fact that these codes are used as inner codes. The block length b of the inner stochastic code can be much smaller than the block length N of the final code. Note however that pseudorandomness needs to hold with respect to channels (and even more complex functions) that have complexity measured as a function of N (which in turn gives a lower bound on b).

We first concentrate on the case where channels are circuits of size N^c (which is the case considered by Guruswami and Smith). This allows setting $k, d, b = O(\log N)$ which in turn means that: we need pseudorandomness against circuits of size $N^c = 2^{\Omega(b)}$, and we are allowed to perform encoding and list-decoding in time $2^{O(b)}$.

However, even with these choices, it seems hard to construct such stochastic codes (no matter what complexity assumption we use). Guruswami and Smith [GS10] were not able to give such explicit constructions. Instead, they settle for a Monte-Carlo construction using the probabilistic method: They describe a probability space over functions $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ for $k, d, b = O(\log N)$ in which a code with the two properties above is chosen with high probability. The description of Enc in this probability space is of length polynomial in N^c, and so this indeed gives a Monte-Carlo construction.[7]

2.5 New constructions of pseudorandom weak inner codes

We observe that we can relax the second property in the definition of stochastic inner codes, and still be able to use them in the framework described in the earlier sections. Specifically, let $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ be a function, we use the following modification of condition (2) above:

2'. We say that Enc is L-**weakly list decodable** with **radius** p if there exists a function Dec such that for every $y \in \{0,1\}^b$, $\mathrm{Dec}(y)$ produces a list of at most L messages $m \in \{0,1\}^k$ that contains all messages $m \in \{0,1\}^k$ for which there exists $r \in \{0,1\}^d$ such that $\delta(y, \mathrm{Enc}(m, r)) \leq p$.

The key difference between "weakly list decodable" and the notion used by Guruswami and Smith (which we will call "strongly list-decodable") is that this definition allows a message m to be encoded to the same value under many different seeds r, whereas the previous definition did not. It turns out that constructing codes with properties 1 and 2′ is significantly simpler than constructing codes with the original properties. Specifically, for the case of inputs and outputs that are of length $O(\log N)$, we give a general transformation that for every $0 < p < \frac{1}{2}$ takes:

- a pseudorandom generator $G : \{0,1\}^{a' \cdot \log N} \to \{0,1\}^{q \log N}$ that is pseudorandom for $\mathcal{C}$,

and converts it into,

- a stochastic code $\mathrm{Enc} : \{0,1\}^{a \log N} \times \{0,1\}^{a' \log N} \to \{0,1\}^{q \log N}$ that is pseudorandom for $\mathcal{C}$, and is L-weakly list decodable with radius p. Here, a, a', q are constants and q is sufficiently larger than a, a' (the exact dependence is $q \geq \frac{a+a'}{1-H(p)-1/(L+1)}$). Furthermore, encoding and list-decoding can be done in time $\mathrm{poly}(N^q)$ times the running time of G.

[7] Note that the obvious approach to checking whether a candidate Enc is pseudorandom against circuits of size N^c requires going over all such circuits which is not feasible in polynomial time.

This transformation works by setting $\mathrm{Enc}(m,r) = E(m) \oplus G(r)$ where $E : \{0,1\}^{a \log n} \to \{0,1\}^{q \log n}$ is a random code. The argument is similar to proofs that random codes are list decodable, and explicitness is achieved by derandomizing the probabilistic argument using $(L+1)$-wise independence, and using brute force decoding. (Here it is crucial that we are allowed to encode and decode in exponential time in the input and output length).[8]

We can use these transformation to obtain stochastic codes that are weakly list-decodable from radius $0 < p < \frac{1}{2}$ and are:

- pseudorandom against size N^c circuits, using the pseudorandom generators of Impagliazzo and Wigderson [IW97] which rely on the assumption that there exists a constant $\beta > 0$ and a problem in $E = \mathrm{DTIME}(2^{O(n)})$ that cannot be solved by circuits of size $2^{\beta \cdot n}$ for every sufficiently large n. This gives Theorem 1.2.

- pseudorandom against space $O(\log n)$ ROBPs, using the pseudorandom generators of Nisan and Zuckerman [NZ96]. This (together with the improvements explained in Section 2.2 and some additional effort that goes into making the reduction implementable by small space ROBPs, explained in Section 6.2) gives Theorem 1.3.

2.6 Inner stochastic codes for $\mathbf{AC}^0$

Our goal is to construct a stochastic code $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ that is weakly-list decodable from radius $p > 0$, and ϵ-pseudorandom against large circuits of constant depth d. We want these codes to have fixed $\mathrm{poly}(n)$ time encoding and decoding. This is because in the final construction, we will choose the block length n to be $N^{0.1}$ (where N is the block length of the final code). This choice will enable fooling circuits of superpolynomial size.

We will use an explicit binary linear code $\mathrm{Enc}_{AG} : \{0,1\}^{d+k} \to \{0,1\}^n$ with constant rate R that decodes pn errors. There are constructions of explicit codes with rate $R > 0$ and $p > 0$, that have the additional property that the relative distance of the dual code is at least p. Such constructions can be obtained by using the Algebraic Geometric codes of Garcia and Stichtenoth [GS96] (that are over constant size alphabets that can be chosen to be a power of two) and viewing them as binary codes (which preserves rate, and decreases relative distance and relative dual distance by a constant factor). A description of these codes appears in a paper by Shpilka [Shp09] (in an appendix attributed to Guruswami), and we elaborate on this result in Section 4.3.

Let G be the $(d+k) \times n$ generator matrix of such codes, and let $G^{(t)}$ denote the $d \times n$ matrix obtained by the first d rows of G, and $G^{(b)}$ denote the bottom $k \times n$ rows of G. For simplicity let us set $k = d$, so that both are linear in n. In the construction of Garcia and Stichtenoth, it can be arranged that $G^{(t)}$ is a generator matrix for a code with similar properties, and in particular the code generated by $G^{(t)}$ has relative dual distance $p > 0$ (we may need to slightly decrease p for this to hold). We define:

$$\mathrm{Enc}(x,r) = \mathrm{Enc}_{AG}(r \circ x) = (r \circ x) \cdot G = r \cdot G^{(t)} + x \cdot G^{(b)}$$

[8] It is this argument that makes the running time of encoding/decoding of our constructions for circuits and online channels, grow with the size of the family of channels. Specifically, encoding and decoding of the inner stochastic code are done by "brute force" and in particular, require running the PRG on all seeds. The number of seeds of a PRG is typically larger than the number of potential distinguishers in the fooled class, and this means that we lose in efficiency, when we try to handle more complex channels.

We note that the dual code to the code defined by $G^{(t)}$ has relative distance p. This means that (the transposed of) $G^{(t)}$ is the parity check matrix of a code with relative distance p, which in turn implies that every pn columns of $G^{(t)}$ are linearly independent. This gives that the distribution $r \cdot G^{(t)}$ for $r \leftarrow U_d$ is pn-wise independent, and implies that for every $x \in \{0,1\}^k$, $\mathrm{Enc}(x, U_d)$ is pn-wise independent. By Braverman's theorem [Bra10] (see also later improvements by [Tal14]) "polylog-wise independence fools AC^0", and in particular pn-independent distributions are pseudorandom for circuits of size $2^{n^{\Omega(1/d)}}$ and depth d.

The code Enc_{AG} is uniquely decodable from pn errors. This immediately gives that Enc it is (strongly) list-decodable with radius p.

Organization of the paper

In Section 3 we give definitions of objects used in out constructions, and the constructions from earlier work that we rely on. In Section 3.2.1 we show how to construct codes against balanced errors. In Section 4 we give precise define several variants of stochastic codes, and give constructions of inner stochastic codes that will be used in the main result. In Section 5 we present the construction of stochastic codes, and restate the theorems from the introduction in a more precise way. In Section 6 we prove the correctness of the construction (and explain how to handle weak classes of channels).

3 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.

3.1 Pseudorandom generators

Definition 3.1 (Pseudorandom generators). *A distribution X on n bits is ϵ-**pseudorandom** for a class $\mathcal{C}$ of functions from n bit 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 $\mathcal{C}$.*

In the sections below, we list the constructions of pseudorandom generators, that we use in this paper. We consider several choices of classes $\mathcal{C}$.

3.1.1 Poly-size circuits

Definition 3.2 (E is hard for exponential size circuits). *We say that E is hard for exponential size circuits if there exists $\beta > 0$ and a language $L \in E = DTIME(2^{O(n)})$ such that for every sufficiently large n, circuits of size $2^{\beta \cdot n}$ fail to compute the characteristic function of L in inputs of length n.*

Theorem 3.3. *[IW97] If E is hard for exponential size circuits then for every constant $c > 1$, there exists a constant $b > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^n$ that is a $\frac{1}{n^c}$-PRG for circuits of size n^c. Furthermore, G is computable in time poly(n^c) (where this polynomial depends on the constant β hidden in the assumption).*

3.1.2 Oblivious read once branching program

Theorem 3.4. *[Nis92, INW94] There exist a constant $a > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{a \cdot \log n \cdot (s + \log(1/\epsilon))} \to \{0,1\}^n$ that is ϵ-pseudorandom for ROBPs of space s. Furthermore, G is computable in time poly(n).*

We also need PRGs with error that is exponentially small in the seed length. In this setup, we only require arbitrary linear stretch.

Theorem 3.5. *[NZ96] For every $b > 1$, there exists a constant $a > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{a \cdot s} \to \{0,1\}^{a \cdot b \cdot s}$ that is a $\frac{1}{2^{-s}}$-PRG for ROBPs of space s. Furthermore, G is computable in time poly(s).*[9]

3.1.3 Constant depth circuits

Theorem 3.6. *[Nis91, NW94, TX13, Tal14] There exists a constant $a > 1$ such that for every constant d, and for every sufficiently large n, there is a $G : \{0,1\}^{(\log(s/\epsilon))^{a \cdot d}} \to \{0,1\}^n$ that is an ϵ-PRG for circuits of size s and depth d. Furthermore, G is computable in time poly(n).*

We will also use Braverman's result that polylog-wise independence fools AC^0.

Theorem 3.7. *[Bra10, Tal14] There exists a constant $a > 1$ such that for every sufficiently large n, every $(\log(s/\epsilon))^{a \cdot d}$-wise independent distribution on n bits is ϵ-pseudorandom for circuits of size s and depth d.*

3.2 Error-Correcting Codes

We give a nonstandard definition of error-correcting codes below. For our purposes it is more natural to define codes in terms of a pair (Enc, Dec) of encoding and decoding algorithms. Different variants are obtained by considering different tasks (decoding, list-decoding, list-recovering) of the decoding algorithms and different types of error vectors.[10]

Definition 3.8 (Codes). *Let k, n, q be parameters and let $Enc : \{0,1\}^k \to (\{0,1\}^{\log q})^n$ be a function. We say that Enc is an encoding function for a code that is:*

- ***decodable** from errors in E (where $E \subseteq (\{0,1\}^{\log q})^n$) if there exists a function $Dec : (\{0,1\}^{\log q})^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $e \in E$, $Dec(Enc(m) \oplus e) = m$. The standard choice of E is the set of all vectors with Hamming weight t, and such codes are said to be* decodable *from t errors.*
- *L-**list-decodable** from errors in E if the function Dec above is allowed to output a list of size at most L that contains m.*

[9] We remark that the construction of [NZ96] can achieve superlinear stretch at the cost of increasing the error. In our application, it is crucial to achieve error that is exponentially small in the seed length, and this is why we state the theorem in this form.

[10] Within this section we use the standard choice of letters of error-correcting codes. However, in later sections many of these letters are reserved to denote other things, and we have to use nonconventional choices.

- (α, ℓ, L)-***list-recoverable*** *if there exists a function Dec which given a list* $T \subseteq \{0,1\}^{\log q}$ *of size at most* ℓ*, outputs a list of size at most* L *containing all* $m \in \{0,1\}^k$ *such that* $\Pr_{i \leftarrow [n]}[Enc(m)_i \in T] \geq \alpha$.[11]

- (α, ℓ, L)-***list-recoverable from a collection*** *if there exists a function Dec which given* n *lists* $T_1, \ldots, T_n \subseteq \{0,1\}^{\log q}$ *of size at most* ℓ*, outputs a list of size at most* L *containing all* $m \in \{0,1\}^k$ *such that* $\Pr_{i \leftarrow [n]}[Enc(m)_i \in T_i] \geq \alpha$.

A code is ***explicit*** *if its encoding and decoding functions are computable in time polynomial in their input and output. The rate of the code is the ratio of the message length and output length of Enc, where both lengths are measured in bits.*

3.2.1 Codes for balanced errors

We will make use of codes for balanced error vectors (as explained in Section 2).

Definition 3.9 (balanced errors)**.** *A string* $e \in \{0,1\}^n$ *is* (b, p, γ)-***balanced*** *if when viewing it as* $e \in (\{0,1\}^b)^{n/b}$ *at most a* γ*-fraction of the* n/b *blocks have hamming weight larger than* $p \cdot b$.

It is not hard to construct codes for balanced errors with rate approaching $1 - H(p)$, using code concatenation. The proof of Theorem 3.10 appears in Section 7.

Theorem 3.10 (codes against balanced errors)**.** *For every constants* $0 < p < 1/2$*,* $\epsilon > 0$*, and* $\gamma \geq \epsilon$ *there are constants* b *and* $L = poly(1/\epsilon)$ *such that for every sufficiently large* n*, there is a code* (Enc, Dec) *with rate* $1 - H(p) - \epsilon$ *that is* L*-list decodable against* (b, p, ϵ)*-balanced strings of length* n*. Moreover the code is explicit (encoding and list-decoding can be performed in time* $poly(n)$*).*

3.2.2 List recoverable codes

We will make use of the following list recoverable code.

Theorem 3.11 (List-recoverable codes)**.** *[Sud97, GS99] There is a constant* $\beta > 0$ *such that for every constants* $\alpha > 0$ *and* $L > 1$*, and every sufficiently large* n*, there is a code* (Enc, Dec) *that is* $(\alpha, \beta \cdot \alpha \cdot L \cdot n, L)$*-list recoverable, has rate* $R \geq \frac{\beta \cdot \alpha}{L}$*, and alphabet size* $q = n^2$.

This follows as Sudan [Sud97] (see also Guruswami and Sudan [GS99]) showed that Reed-Solomon codes are list-recoverable from a collection. Given a code Enc that is list-recoverable from a collection, $\text{Enc}'(x)_i = (\text{Enc}(x), i)$ gives a code that is list recoverable, while increasing the alphabet size. This is why we have the alphabet size of $q = n^2$ (and not $q = n$) for a Reed Solomon code. This idea is also implicitly used by Guruswami and Smith [GS10].

3.3 Averaging Samplers

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

[11] This is a less standard notion of list-recoverability, and the more standard notion referred to as "list-recoverable" is what we call "list-recoverability from a collection" in the next item.

Definition 3.12 (Averaging Samplers). *A function* $Samp : \{0,1\}^n \to (\{0,1\}^m)^t$ *is an* (ϵ,δ)-***Sampler*** *if for every* $f : \{0,1\}^m \to [0,1]$, $\Pr_{(z_1,\ldots,z_t)\leftarrow Samp(U_n)}[|\frac{1}{t}\sum_{i\in[t]} f(z_i) - \frac{1}{2^m}\sum_{x\in\{0,1\}^m} f(x)| > \epsilon] \leq \delta$. *A sampler has* ***distinct samples*** *if for every* $x \in \{0,1\}^n$, *the elements in* $Samp(x)$ *are distinct.*

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

Theorem 3.13. *For every sufficiently large* m *and every* $\epsilon \geq \delta > 0$ *there is a* (ϵ,δ)*-sampler,* $Samp : \{0,1\}^{O(m+\log(1/\delta)\cdot poly(1/\epsilon))} \to (\{0,1\}^m)^t$ *for any* $t \geq poly(1/\epsilon)\cdot\log(1/\delta)$. *Furthermore,* $Samp$ *is computable in time* $poly(m, 1/\epsilon, \log(1/\delta))$.

3.4 Almost t-wise permutations

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

Definition 3.14 (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* $\pi_s(x) = \pi(s,x)$ *is a permutation over* $[n]$.
- *For every* $x_1,\ldots,x_t \in [n]$, *the random variable* $R = (R_1,\ldots,R_t)$ *defined by* $R_i = \pi(s,x_i)$: $s \leftarrow U_d$, *is* ϵ*-close to* t *uniform samples without repetition from* $[n]$.

Theorem 3.15. *[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,* π *is computable in polynomial time.*

4 Inner Stochastic codes

As explained in Sections 2.4 and 2.5, the construction will rely on an "inner stochastic code". We now give a formal definition of the properties required from these codes. This definition formalizes the looser description given in Section 2.

Definition 4.1. *Let* k,n,q *be parameters and let* $Enc : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ *be a function. We say that* Enc *is an encoding function for a stochastic code that is:*

- L-***weakly list-decodable*** *with* ***radius*** p *if there exists a function* Dec *such that for every* $y \in \{0,1\}^n$, $Dec(y)$ *produces a list of at most* L *messages that contains all messages* $m \in \{0,1\}^k$ *for which there exists* $r \in \{0,1\}^d$ *such that* $\delta(y, Enc(m,r)) \leq p$.
- *We replace "weakly" with "****strongly****" if* Dec *is required to produce a list of at most* L *pairs* (m,r) *that contains all pairs* $(m,r) \in \{0,1\}^k \times \{0,1\}^d$ *such that* $\delta(y, Enc(m,r)) \leq p$.
- ϵ-***pseudorandom*** *for a class* $\mathcal{C}'$ *of functions from* n *bits to one bit, if for every message* $m \in \{0,1\}^k$, $C(m,U_d)$ *is* ϵ*-pseudorandom for* $\mathcal{C}'$.

If we do not mention whether the code is weakly or strongly list-decodable, then we mean "weakly". In the remainder of this section we give explicit constructions of "inner stochastic codes" for the various channel classes that we consider. We start with a general transformation that transforms a PRG into an inner stochastic code.

4.1 PRGs give inner stochastic codes

We give a general transformation that given a PRG with:

- A seed length that is logarithmic in the complexity of the channel.
- Sufficiently large linear stretch as a function of p.

Produces a stochastic code that:

- Inherits the logarithmic seed length and pseudorandomness properties of the PRG.
- Is able to encode a string of length logarithmic in the complexity of the channel.
- Is L-weakly list decodable from radius p where L is a constant.
- Has encoding and decoding running in time polynomial in the complexity of the channel, and the running time of the PRG.

This transformation is formally stated in the next theorem. We need the following definition that formally defines the action (which we call "xored-restriction") of restricting functions to a subset of the input, and negating some of the remaining input bits. The complexity classes that we consider in this paper (AC^0, P/poly, logspace ROBPs) are all closed under xored restriction. (This is also the case for any natural nonuniform complexity class).

Definition 4.2 (xored restriction). *We say that a function C' over n' bits is an xored-restriction of a function C over n bits if there exist strings $y \in \{0,1\}^{n'}$, $a \in \{0,1\}^{n-n'}$ and a set $S \subseteq [n]$ of size n' such that for every input x', $C'(x') = C(x)$, where x is an n bit string obtained by "filling" the indices in S with $x' \oplus y$, and the indices outside of S with a.*

Theorem 4.3 (inner stochastic code from PRG). *Let $\mathcal{C}, \mathcal{C}'$ be classes of functions, and $a > 0$, $b > 0$, $L \geq 1$ and $0 \leq p < \frac{1}{2}$ be constants such that $(1 - \frac{1}{L+1}) > H(p)$, and assume that n is sufficiently large. Let $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ be an ϵ-PRG for class $\mathcal{C}'$ such that $q \geq \frac{a+b}{1-H(p)-\frac{1}{L+1}}$. There is a stochastic code (Enc_{SC}, Dec_{SC}) where $Enc_{SC} : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ that is:*

- *L-weakly list decodable from radius p.*
- *If every xored restriction of $\mathcal{C}$ is in $\mathcal{C}'$ then Enc_{SC} is ϵ-pseudorandom for $\mathcal{C}$.*
- *The algorithms Enc_{SC}, Dec_{SC} are computable in time poly$(n^{q \cdot L})$ given oracle access to G. (In particular, the code is explicit if G runs in time poly(n)).*

Proof. The code will be a combination of two functions, $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ and $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$, and we will have: $\mathrm{Enc}_{SC}(x, r) = E(x) \oplus G(r)$.

We will use a probabilistic construction (similar to that used to show existence of capacity achieving, binary list decodable codes) which we later derandomize using $(L+1)$-wise independence.

Claim 4.4. *Let $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ be chosen at random, so that the random variables $(E(x))_{x \in \{0,1\}^{a \cdot \log n}}$ are $(L+1)$-wise independent. Then, with positive probability, $Enc_{SC}(x, r) = E(x) \oplus G(r)$ is L-weakly list decodable from radius p.*

Proof. (of claim) Given $y \in \{0,1\}^{q \log n}$, we use $B(y,p)$ to denote the ball of radius $p \cdot (q \cdot \log n)$ centered at $y \in \{0,1\}^{q \log n}$. For every $x \in \{0,1\}^{a \log n}$ and $y \in \{0,1\}^{q \log n}$ we define a random variable indicator

$$Z^{x,y} = \begin{cases} 1 & \text{if } \exists r \in \{0,1\}^{b \cdot \log n} \text{ such that, } \mathrm{Enc}_{SC}(x,r) \in B(y,p) \\ 0 & \text{otherwise} \end{cases}$$

We have that:

$$\begin{aligned} \Pr[Z^{x,y} = 1] &\leq 2^{b \log n} \cdot \frac{2^{H(p) \cdot q \cdot \log n}}{2^{q \log n}} \\ &\leq 2^{\log n \cdot (b + q(H(p)-1))} \end{aligned}$$

Given a tuple $x_1, \ldots, x_{L+1} \in \{0,1\}^{a \log n}$ and $y \in \{0,1\}^{q \log n}$, let $B^{x_1,\ldots,x_{L+1},y}$ be the "bad event" that the $L+1$ points $x_1, \ldots, x_{L+1}$ all have seeds of G that make them land in the ball of y, namely:

$$B^{x_1,\ldots,x_{L+1},y} = \left\{ \forall i \in [L+1], \exists r \in \{0,1\}^{b \log n} \text{ such that } E(x_i) \oplus G(r) \in B(y,p) \right\}.$$

The random variables $E(x_1), \ldots, E(x_{L+1})$ are independent, and therefore,

$$\Pr[B^{x_1,\ldots,x_{L+1},y}] = \prod_{i=1}^{L+1} \Pr[Z^{x,y} = 1] \leq 2^{(\log n) \cdot (b + q(H(p)-1))(L+1)}$$

Note that $\mathrm{Enc}_{SC}(x,r) = E(x) \oplus G(r)$ is L-weakly list decodable from radius p, if and only if E does not belong to $B^{x_1,\ldots,x_{L+1},y}$ for all choices of $x_1, \ldots, x_{L+1} \in \{0,1\}^{a \cdot \log n}$ and $y \in \{0,1\}^{q \cdot \log n}$. Therefore, by a union bound, the probability that we don't obtain an L-weakly list decodable code from radius p, is at most:

$$\begin{aligned} \sum_{x_1,\ldots,x_{L+1},y} \Pr[B^{x_1,\ldots,x_{L+1},y}] &\leq 2^{q \log n} \cdot \binom{2^{a \log n}}{L+1} \cdot 2^{(\log n) \cdot (b + q(H(p)-1))(L+1)} \\ &< 2^{(\log n) \cdot (q + a(L+1) + (b + q(H(p)-1))(L+1))} \end{aligned}$$

Thus, if $q \geq \frac{a+b}{1 - \frac{1}{L+1} - H(p)}$, then the probability is less than one, and there exists an L-weakly list decodable code from radius p.[12]

□

Given oracle access to a candidate function $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ and to $G : \{0,1\}^{b \log n} \to \{0,1\}^{\log q}$ we can check whether E induces a code with the required properties in time $\mathrm{poly}(n^q)$.

It is standard that there are constructions of $2^{a \log n} = n^a$ random variables that are $(L+1)$-wise, and each variable is uniform over $\{0,1\}^{q \log n}$, that can be sampled using only $(L+1) \cdot q \log n$ random bits. Therefore, in time $\mathrm{poly}(n^{L \cdot q})$ we can go over all candidate E's, and find one which induces an L-weakly list decodable from radius p.

Once we find a good function E we are guaranteed that Enc_{SC} is ϵ-pseudorandom for $\mathcal{C}$.

[12] We remark that it is also possible to extend proofs that random linear codes achieve list decoding capacity to show that we can obtain a linear code E that yields a good code Enc_{SC}.

Claim 4.5. *For every $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$, the function $Enc_{SC}(x,r) = E(x) \oplus G(r)$ is ϵ-pseudorandom for $\mathcal{C}$.*

Proof. Otherwise, there exists $x' \in \{0,1\}^{a \log n}$ and a function $C \in \mathcal{C}$ that distinguishes $\mathrm{Enc}_{SC}(x', U_{b \log n}) = E(x') \oplus G(U_{b \log n})$ from uniform. This means that there is an xored restriction C' of C that distinguishes $G(U_{b \log n})$ from uniform, and this is a contradiction. □

Finally, it remains to justify the claim about the decoding procedure. Given a string $y \in \{0,1\}^{q \log n}$, the decoding algorithm will use brute force to go over all $(x,r) \in \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n}$, and check for each whether $\delta(\mathrm{Enc}(x,r), y) \leq p$. By the L-weakly list decodable property, there will be at most L distinct values of x. The decoding complexity is $O(2^{a \log n} \cdot 2^{b \log n}) = \mathrm{poly}(n^{a+b})$ with oracle access to G. □

4.2 Inner Stochastic codes for circuits and ROBPs

By plugging in the pseudorandom generators from Theorems 3.3 and Theorem 3.5 in Theorem 4.3. We immediately obtain the following stochastic codes (that will be used in the construction).

Theorem 4.6 (inner stochastic code for poly-size circuits)**.** *If E is hard for exponential size circuits then for every constant $0 \leq p < \frac{1}{2}$, $c > 1$ and $a > 0$ there exist constants L, b, q such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ is:*

- *L-weakly list decodable from radius p.*
- *$\frac{1}{n^c}$-pseudorandom for size n^c circuits.*

Furthermore, the code is explicit. Specifically, Enc, Dec are computable in time $\mathrm{poly}(n^c)$, where the polynomial depends on p, a and the constant $\beta > 0$ hidden in the hardness assumption.

Theorem 4.7 (inner stochastic code for online channels)**.** *For every constant $0 \leq p < \frac{1}{2}$, $c > 1$ and $a > 0$ there exist constants L, b, q such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ is:*

- *L-weakly list decodable from radius p.*
- *$\frac{1}{n^c}$-pseudorandom for space $c \log n$ ROBPs.*

Furthermore, the code is explicit. Specifically, Enc, Dec are computable in time $\mathrm{poly}(n^c)$ where the polynomial depends on p, a.

4.3 Inner stochastic codes for AC^0 channels

In this section we give a construction of inner stochastic codes for circuits of constant depth. This construction has the advantage that the encoding and decoding of the inner stochastic code run in fixed polynomial time, and do not depend on the size or depth of the circuit family.

Theorem 4.8 (inner stochastic code for AC^0)**.** *There exist constants $p > 0$, $R > 0$ and $a > 1$ such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{Rn} \times \{0,1\}^{Rn} \to \{0,1\}^n$ that is:*

- 1-*strongly list decodable from radius* p.
- $2^{-n^{\frac{1}{ad}}}$-*pseudorandom for circuits of size* $2^{n^{\frac{1}{ad}}}$ *and depth* d.

Furthermore, the code is explicit. Specifically, Enc, Dec *are computable in time* $poly(n)$*, for a fixed universal polynomial (only the choice of what* n *is sufficiently large depends on the constants).*

Proof. The theorem will follow from the following claim.

Claim 4.9. *There exist constants* $p > 0$, $R > 0$ *such that for every sufficiently large* n*, there is a* $2Rn \times n$ *matrix* $G^{(n)}$ *such that:*

- $G^{(n)}$ *is a generator matrix for a binary linear* $[n, 2Rn]$*-code that is decodable from* pn *errors.*
- *Let* $G_t^{(n)}$ *be the* $Rn \times n$ *matrix obtained by taking the first* Rn *rows of* $G^{(n)}$*.* $G_t^{(n)}$ *is a generator matrix for a binary linear* $[n, Rn]$*-code such that its dual code has distance larger than* pn*.*
- *The code* (Enc, Dec) *that is defined by* $G^{(n)}$ *is explicit (and in particular* $G^{(n)}$ *can be constructed in time* $poly(n)$*).*

Proof. (of claim) It is sufficient to prove the lemma for codes with alphabet size 2^s for some constant s (rather than for binary codes). This is because, such codes can be viewed as binary codes (in a natural way) and this viewpoint preserves rate, and decreases relative distances (or the fraction of errors that can be decoded) by a constant factor of $1/s$. We therefore focus on proving the claim for codes with alphabet size that is constant and a power of two.

There are codes (based on algebraic geometric codes) over constant size alphabet where the size can be a power of two, that have: constant rate, can be explicitly encoded and decoded from a constant fraction of errors, and furthermore have a positive relative dual distance. Such codes follow from the work of Garcia and Stichtenoth [GS96] and a self contained summary is presented in [Shp09] (the summary is in an appendix written by Guruswami). Theorem 24 in the appendix contains a precise statement on the existence of such codes.

An inspection of the proof reveals that this argument can also be used to obtain two explicit linear codes $C_t \subseteq C$ with the properties above. More specifically, by varying the parameters in the proof, there exist constants $R > 0$ and $p > 0$ such that for sufficiently large n, C_t has constant rate $R > 0$, C has rate $2R > 0$ and both codes have the properties listed above, namely: C_t (resp. C) can be efficiently decoded from $p \cdot n$ errors (for some $p > 0$) and both codes have dual distance $p \cdot n$. Loosely speaking, this follows as one can perform the argument once to obtain one code C_t, and then increase the dimension, to give a code C such that $C_t \subseteq C$ with the same properties.

The matrix G_t will be the generator matrix of C_t and it can be easily extended to a generator matrix G of C. □

We now observe that the claim implies the theorem. The stochastic code $Enc : \{0,1\}^{Rn} \times \{0,1\}^{Rn} \to \{0,1\}^n$ is defined as follows: Given $x, r \in \{0,1\}^{Rn}$, let y be the concatenation $y = r \circ x$ and $Enc(x, r) = y \cdot G$.

This code is 1-strongly list decodable from radius p by the decoding properties of the code generated by G. More precisely, given $z \in \{0,1\}^n$, we can decode to a unique message $y \in \{0,1\}^{2Rn}$ that has hamming distance at most pn from z, and this message $y = (x, r)$ can be found efficiently.

We now show the pseudorandomness of Enc. Let G_b denote the bottom Rn rows of G (and recall that G_t denotes the top Rn rows of G). For every $x, r \in \{0,1\}^{Rn}$,

$$\mathrm{Enc}(x, r) = (r \circ x) \cdot G = r \cdot G_t + x \cdot G_b.$$

The generator matrix G_t generates a code with dual distance at least pn. This means that transposed matrix is the parity matrix of the dual code. The fact that the dual code has distance larger than pn, implies that every pn columns of G_t are linearly independent. This gives that the distribution $r \cdot G_t$ for $r \leftarrow U_{Rn}$ is pn-wise independent, and implies that for every $x \in \{0,1\}^{Rn}$, $\mathrm{Enc}(x, U_{Rn})$ is pn-wise independent. Braverman [Bra10] (and later improvements by Tal [Tal14]) (See Theorem 3.7) showed that t-wise independent distributions are ϵ-pseudorandom for circuits of size s and depth d, if $t \geq (\log \frac{s}{\epsilon})^{c \cdot d}$ for some constant c. This gives that there exists a constant $a > 1$ such that $\mathrm{Enc}(x, U_{Rn})$ is $2^{-n^{\frac{1}{ad}}}$-pseudorandom for circuits of size $2^{n^{\frac{1}{ad}}}$ and depth d, as required. □

5 The construction of stochastic codes

In this section we give the construction of the stochastic code. Our construction imitates that of Guruswami and Smith [GS10] (with the modifications explained in Section 2). We start with introducing some notation.

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

- *Given strings $c_{data} \in \{0,1\}^{N_{data}}$ and $c_{ctrl} \in \{0,1\}^{N_{ctrl}}$ we define an N bit string c denoted by $(c_{data}, c_{ctrl})^I$ as follows: We think of c_{data}, c_{ctrl}, c as being composed of blocks of length b (that is $c_{data} \in (\{0,1\}^b)^{n_{data}}$, $c_{ctrl} \in (\{0,1\}^b)^{n_{ctrl}}$ and $c \in (\{0,1\}^b)^n$). We enumerate the indices in $[n] \setminus I$ by $j_1, \ldots, j_{n_{data}}$ and set $c_\ell = \begin{cases} (c_{ctrl})_k & \text{if } \ell = i_k \text{ for some } k; \\ (c_{data})_k & \text{if } \ell = j_k \text{ for some } k \end{cases}$*
- *Given a string $c \in \{0,1\}^N$ (which we think of as $c \in (\{0,1\}^b)^n$) we define strings c^I_{data}, c^I_{ctrl} by $c_{ctrl} = c|_I$ and $c_{data} = c|_{[n]\setminus I}$, (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.

Permuting strings. Our construction will also use permutations to permute strings as follows:

Definition 5.2. *Given a string $v \in \{0,1\}^N$ and a permutation $\pi : [N] \to [N]$. Let $\pi(v)$ denote the string $v' \in \{0,1\}^N$ with $v'_i = v_{\pi(i)}$.*

Figure 1: Parameters and ingredients for stochastic code

Parameters:

- N - The length (in bits) of the codeword. (Throughout we assume that N is sufficiently large). Other parameters are either constants or chosen as a function of N.
- p - The fraction of errors we need to recover from. This is a constant.
- $\mathcal{C}'$ - A class of functions (typically slightly stronger than the class of channels we allow).
- $0 < \epsilon < \frac{1}{2} - p$ - We want rate $R = 1 - H(p) - \epsilon$, meaning that messages have length RN. ϵ is a constant.
- b - We will divide the N output bits to $n = N/b$ blocks of length b, where $2 \log N \leq b \leq N^{1/10}$ is a function of N that will be chosen later on. This implies $n \geq N^{0.9}$.
- $\nu \geq 2^{-\sqrt{N}}$ - A bound on the failure probability of decoding (can be chosen as a function of N).

Internal parameters:

- Blocks will be of two kinds: "control" and "data". We set $n_{\text{ctrl}} = \epsilon \cdot n$ and $n_{\text{data}} = n - n_{\text{ctrl}}$ so that $n = n_{\text{ctrl}} + n_{\text{data}}$. Let $N_{\text{ctrl}} = b \cdot n_{\text{ctrl}}$ and $N_{\text{data}} = b \cdot n_{\text{data}}$. So that $N = N_{\text{ctrl}} + N_{\text{data}}$.
- Let $\alpha > 0$ be a sufficiently small constant that will be chosen later.
- Let $\ell_{\text{ctrl}} = N^{0.8}$ and $\ell'_{\text{ctrl}} = \ell_{\text{ctrl}}/3$.

Ingredients that depend on the choice of channel class: We assume that we are given:

- A stochastic code $\text{Enc}_{SC} : \{0,1\}^{2 \log n_{\text{ctrl}}} \times \{0,1\}^{\ell'_{\text{SC}}} \to \{0,1\}^b$ that is ϵ_{SC}-pseudorandom for $\mathcal{C}'$ (for $\epsilon_{SC} = \frac{\nu}{10 \cdot n_{\text{ctrl}}}$) and is L_{SC}-weakly list decodable from radius $p + \epsilon$. We require that L_{SC} is a constant, and $\ell'_{\text{SC}} \leq N$.
- An ϵ_{PRG}-PRG $\text{PRG} : \{0,1\}^{\ell'_{\text{ctrl}}} \to \{0,1\}^{N_{\text{data}}}$ for $\mathcal{C}'$, for $\epsilon_{PRG} = \frac{1}{10} \cdot \nu$.

Other Ingredients:

- A code $\text{Enc}_{\text{balanced}} : \{0,1\}^{RN} \to \{0,1\}^{N_{\text{data}}}$ with an algorithm $\text{Dec}_{\text{balanced}}$ that performs L_{balanced}-list decoding from $(b', p + \alpha, \alpha)$-balanced errors. By Theorem 3.10 we have an explicit construction with rate $R' \geq 1 - H(p + \alpha) - \alpha$ where b' and L_{balanced} are large constants (chosen as a function of the constants α and p). By choosing a sufficiently small $\alpha > 0$ we indeed have $R' \geq RN/N_{\text{data}} = R/(1 - \epsilon)$.
- A code $\text{Enc}_{LR} : \{0,1\}^{\ell_{\text{ctrl}}} \to (\{0,1\}^{2 \log n_{\text{ctrl}}})^{n_{\text{ctrl}}}$ that is $(\frac{\epsilon^2}{100}, L_{SC} \cdot n, L_{LR})$-list recoverable. Note that $L_{SC} \cdot n = \frac{L_{SC}}{\epsilon} \cdot n_{\text{ctrl}}$. By Theorem 3.11 we can obtain such a code with constant rate $R' > 0$ for some constant L_{LR} (these two constants depend on ϵ). The rate we allow for Enc_{LR} above is $\frac{\ell_{\text{ctrl}}}{2 \log n_{\text{ctrl}} \cdot n_{\text{ctrl}}} \leq \frac{N^{0.8}}{\epsilon \cdot n} = o(1) \leq R'$.
- A $(2^{-N^{0.6}}, N^{0.6})$-wise permutation $\pi : \{0,1\}^{\ell'_{\text{ctrl}}} \times [N_{\text{data}}] \to [N_{\text{data}}]$. By Theorem 3.15 we have an explicit construction with seed length $N^{0.7} \leq \ell'_{\text{ctrl}}$.
- An $(2^{-N^{0.6}}, min(\frac{\alpha}{100}, \frac{\epsilon^2}{100}))$-sampler with distinct samples $\text{Samp} : \{0,1\}^{\ell'_{\text{ctrl}}} \to [n]^{n_{\text{ctrl}}}$. By Theorem 3.13 we have an explicit construction with seed length $O(N^{0.7}) \leq \ell'_{\text{ctrl}}$ and $N^{0.7} \leq \epsilon \cdot n = n_{\text{ctrl}}$ samples.

Description of the construction. Our construction is described in detail in the three figures below. The choice of parameters and ingredients is described in Figure 1. The encoding algorithm

is described in Figure 2, and the list-decoding algorithm is described in Figure 3. We state a general theorem that summarizes the correctness of the construction and will be used to prove Theorems 1.2, 1.3, 1.4.

Figure 2: Encoding algorithm for stochastic code

Input:

- A message $m \in \{0,1\}^{RN}$.
- A "random part" r for the stochastic encoding that consists of a string $s = (s_{\text{samp}}, s_\pi, s_{\text{PRG}})$ where $s_{\text{samp}}, s_\pi, s_{\text{PRG}} \in \{0,1\}^{\ell'_{\text{ctrl}}}$ so that $s \in \{0,1\}^{\ell_{\text{ctrl}}}$, and $r_1, \ldots, r_{n_{\text{ctrl}}} \in \{0,1\}^{\ell'_{\text{SC}}}$.

Operation:

Determine control blocks: Apply $\text{Samp}(s_{\text{samp}})$ to generate $I = \{i_1, \ldots, i_{n_{\text{ctrl}}}\} \subseteq [n]$. 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 $x = \text{Enc}_{\text{balanced}}(m)$.
- 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, $y = \pi_{s_\pi}^{-1}(x) = \pi_{s_\pi}^{-1}(\text{Enc}_{\text{balanced}}(m))$.
- Mask y using PRG. That is, $c_{\text{data}} = y \oplus \text{PRG}(s_{\text{PRG}}) = \pi_{s_\pi}^{-1}(\text{Enc}_{\text{balanced}}(m)) \oplus \text{PRG}(s_{\text{PRG}})$.

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:

- Encode s by $z = \text{Enc}_{LR}(s)$. This is a string composed of n_{ctrl} blocks of length $2 \log n_{\text{ctrl}}$.
- Use Enc_{SC} as an "inner code" to encode blocks of z using the randomness $r_1, \ldots, r_{n_{\text{ctrl}}}$. That is, $(c_{\text{ctrl}})_j = \text{Enc}_{SC}(z_j, r_j) = \text{Enc}_{SC}(\text{Enc}_{LR}(s)_j, r_j)$.

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, $c = (c_{\text{data}}, c_{\text{ctrl}})^I$.

Correctness of the construction. Let $\mathcal{C}$ be a class of channels $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN errors. We now show that if the ingredients $\text{PRG}, \text{Enc}_{SC}$ are pseudorandom for a class $\mathcal{C}'$ that is sufficiently stronger than $\mathcal{C}$, then the decoding algorithm of Figure 3 succeeds with high probability. This is stated precisely, in the next theorem, which uses the notion of "xored restrictions" defined in Definition 4.2. (We remind the reader that nonuniform complexity classes as the ones we consider in this paper, are closed under xored restrictions).

Theorem 5.3 (Correctness of construction). *For every constants $0 \leq p < \frac{1}{2}$ and $0 < \epsilon < \frac{1}{2} - p$ there exists a constants $L = L_{LR} \cdot L_{balanced}$ such that for every sufficiently large N the following holds:*

- *Let $\mathcal{C}$ be a class of functions $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN errors. For a channel $C \in \mathcal{C}$, let $E_C(z) = z \oplus C(z)$ denote the error vector (of Hamming weight at most*

Figure 3: List-decoding algorithm for stochastic code

Input: A "received word" $c' \in \{0,1\}^{RN}$.

Operation:

Determine few candidates for control information:

Decode inner code SC: For every $i \in [n]$ apply the list decoding algorithm of SC to generate a size L_{SC} list, $\mathrm{List}_i = \mathrm{Dec}_{SC}(c'_i)$ (here c'_i is the i'th block of c'). Let $\mathrm{List}_{SC} = \cup_{i\in[n]} List_i$.

Decode outer code LR: Apply the list recovering algorithm of LR to generate a size L_{LR} list, $\mathrm{List}_{\mathrm{ctrl}} = \mathrm{Dec}_{LR}(List_{SC})$.

Use each control candidate s to decode data: For each $s = (s_{\mathrm{samp}}, s_\pi, s_{\mathrm{PRG}}) \in \mathrm{List}_{\mathrm{ctrl}}$ (recall that there are L_{LR} of them) we produce a list List_s of L_{balanced} candidate messages. Our final output list will be the union of these lists.

Determine control blocks: Apply $\mathrm{Samp}(s_{\mathrm{samp}})$ to generate $I = \{i_1, \ldots, i_{n_{\mathrm{ctrl}}}\}$. Compute $c'_{\mathrm{data}} = (c')^I_{\mathrm{data}}$.

Unmask PRG: Compute $y'_{\mathrm{data}} = c'_{\mathrm{data}} \oplus \mathrm{PRG}(s_{\mathrm{PRG}})$.

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 $x' = \pi_{s_\pi}(y'_{\mathrm{data}}) = \pi_{s_\pi}(c'_{\mathrm{data}} \oplus \mathrm{PRG}(s_{\mathrm{PRG}}))$.

Decode data: Compute $\mathrm{List}_s = \mathrm{Dec}_{\mathrm{balanced}}(x')$.

Merge lists: The final output is $\mathrm{List} = \bigcup_{s\in\mathrm{List}_{\mathrm{ctrl}}} \mathrm{List}_s$.

pN) induced by the channel.

- *Let $\mathcal{C}'$ be the class of all functions that output one bits, and are xored restrictions of functions of the form $f(z) = A(E_C(z)))$ where A is either,*
  - *a size N^{c_0}, depth d_0 circuit, for some universal constants c_0, d_0.*
  - *a space $\eta_0 \cdot \log 1/\nu \cdot \log N$ ROBP, for some universal constant $\eta_0 > 0$ (which gives space $O(\log N)$ if ν is inverse polynomial in N).*

If the parameters and ingredients are chosen as in Figure 1, then the stochastic code (Enc, Dec) specified in Figures 2, 3, satisfies:

- *It has rate $R \geq 1 - H(p) - \epsilon$.*
- *It is L-list decodable with success probability $1 - \nu$ for channels in $\mathcal{C}$, where $L = poly(1/\epsilon)$ is a constant.*
- *There exist a universal polynomial $P(\cdot)$ such that:*
  - *The function Enc can be computed in $DTIME^{PRG, Enc_{SC}}(P(N))$ (and is therefore explicit if PRG, Enc_{SC} are explicit).*
  - *The function Dec can be computed in $DTIME^{PRG, Dec_{SC}}(P(N))$ (and is therefore explicit if PRG, Dec_{SC} are explicit).*

5.1 Choosing ingredients and parameters for specific channel families

We now put everything together and choose pseudorandom generators and inner stochastic codes for poly-size circuits, online logspace, and AC^0.

5.1.1 Poly-size circuit channels

Here we use the pseudorandom generator of Impagliazzo and Wigderson [IW97] (that requires the assumption that E is hard for exponential size circuits). This PRG has logarithmic seed length, and can be used as PRG, as well as the pseudorandom generator that is transformed into an inner stochastic code Enc_{SC} (as done in Theorem 4.6). The precise statement and parameter choices appear below:

Theorem 5.4 (explicit codes for poly-size channels)**.** *Assume that E is hard for exponential size circuits. For every constants $0 \leq p < \frac{1}{2}$, $\epsilon > 0$, and $c > 1$ and for every sufficiently large N:*

- *Let $\nu = N^{-c}$.*
- *Let $\mathcal{C}$ be the class of all circuits $C : \{0,1\}^N \to \{0,1\}^N$ of size N^c that induce at most pN-errors.*
- *Let $\mathcal{C}'$ be the class of all size N^{2c} circuits that output one bit (this includes circuits for all input lengths up to N). Here, we assume w.l.o.g. that c is sufficiently large so that in time N^{2c} we can compose size N^c computations with fixed polynomial size computations.*
- *Let $(\mathrm{Enc}_{SC}, \mathrm{Dec}_{SC})$ and the block length b be determined by Theorem 4.6. Specifically, let $b = q \cdot \log N$ for a sufficiently large constant q, guaranteed by Theorem 4.6 so that we get that $\mathrm{Enc}_{SC} : \{0,1\}^{2\log n_{ctrl} \leq 2\log N} \times \{0,1\}^{\ell'_{SC} = O(\log N)} \to \{0,1\}^b$ is L_{SC}-weakly list decodable from radius $p+\alpha$ for a sufficiently large constant L_{SC} (chosen as a function of p), and furthermore, Enc_{SC} is $N^{-(c+1)}$-pseudorandom for $\mathcal{C}'$. (Note that $N^{-(c+1)} \leq \nu/10 \cdot n_{ctrl}$ as required).*
- *Let $PRG : \{0,1\}^{O(\log N)} \to \{0,1\}^{N_{data}}$ be an $N^{-(c+1)}$-PRG for $\mathcal{C}'$ from Theorem 3.3, and note that the seed length is smaller than ℓ'_{ctrl}, and $N^{-(c+1)} \leq \nu/10$ as required.*

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) specified in the figures has rate $1 - H(p) - \epsilon$, and is $L = O(1)$-list decodable with success probability $1 - N^{-c}$ against channels in $\mathcal{C}$. Furthermore, Enc, Dec are computable in time $\mathrm{poly}(N^c)$ where the polynomial depends on p, and on the constant $\beta > 0$ hidden in the assumption.

5.1.2 Online logspace channels

Here we use the pseudorandom generator of Nisan [Nis92]. This PRG has seed length that is poly-logarithmic, and can be used as PRG. However, it unsuitable to serve in the construction of inner stochastic codes. This is because the dependence of the seed length on the error, does not allow linear stretch with error that is exponentially small in the seed length. Instead, we use the pseudorandom generator of Nisan and Zuckerman [NZ96], that has these properties and can be transformed into an inner stochastic code Enc_{SC} (as done in Theorem 4.7). The precise statement and parameter choices appear below:

Theorem 5.5 (explicit codes for online logspace channels). *For every constants $0 \leq p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$ and for every sufficiently large N:*

- *Let $\nu = N^{-c}$.*
- *Let $\mathcal{C}$ be the class of all space $c \log N$ ROBPs $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN-errors.*
- *Let $\mathcal{C}'$ be the class of all space $2c \log N$ ROBPs that output one bit (this includes ROBPs for all input lengths up to N). Here we assume w.l.o.g. that c is sufficiently large so that an ROBP of space $2c \log N$ can compose space $c \log N$ online computation with $c_0 \log N$ online computation, for any fixed c_0.*
- *Let (Enc_{SC}, Dec_{SC}) and the block length b be determined by Theorem 4.7. Specifically, let $b = q \cdot \log N$ for a sufficiently large constant q, guaranteed by Theorem 4.7 so that we get that $Enc_{SC} : \{0,1\}^{2 \log n_{ctrl} \leq 2 \log N} \times \{0,1\}^{\ell'_{SC} = O(\log N)} \to \{0,1\}^b$ is L_{SC}-weakly list decodable from radius $p + \alpha$ for a sufficiently large constant L_{SC} (chosen as a function of p), and furthermore, Enc_{SC} is $N^{-(c+1)}$-pseudorandom for $\mathcal{C}'$. (Note that $N^{-(c+1)} \leq \nu/10 \cdot n_{ctrl}$ as required).*
- *Let $PRG : \{0,1\}^{O(\log^2 N)} \to \{0,1\}^{N_{data}}$ be an $N^{-(c+1)}$-PRG for $\mathcal{C}'$ from Theorem 3.4, and note that the seed length is smaller than ℓ'_{ctrl}, and $N^{-(c+1)} \leq \nu/10$ as required.*

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) specified in the figures has rate $1 - H(p) - \epsilon$, and is $L = O(1)$-list decodable with success probability $1 - N^{-c}$ against channels in $\mathcal{C}$. Furthermore, Enc, Dec are computable in time $poly(N^c)$ where the polynomial depends on p.

5.1.3 Constant depth channels

Here we use the pseudorandom generator of Nisan [Nis91]. This PRG has seed length that is subpolynomial for any fixed constant depth d, and can be used as PRG. We use the construction of inner stochastic codes given in Theorem 4.8 for Enc_{SC}. This construction only works for $p < p_0$ for some $p_0 > 0$ and this requirement is inherited by our final theorem. The precise statement and parameter choices appear below:

Theorem 5.6 (explicit codes for constant depth channels). *There exists a constant $p_0 > 0$, $d_0 > 1$ and $a > 0$ such that for every constants $0 \leq p < p_0$, $\epsilon > 0$, $d > 1$ and for every sufficiently large N:*

- *Let $\nu = 2^{-N^{\frac{1}{ad}}}$.*
- *Let $\mathcal{C}$ be the class of circuits $C : \{0,1\}^N \to \{0,1\}^N$ of size $2^{N^{\frac{1}{ad}}}$ and depth d that induce at most pN-errors.*
- *Let $\mathcal{C}'$ be the class of all size $2^{2N^{\frac{1}{ad'}}}$ and depth $d' = d + d_0$ circuits that output one bit (this includes circuits for all input lengths up to N).*
- *Let $b = N^{1/10}$ and let (Enc_{SC}, Dec_{SC}) be determined from Theorem 4.8. Specifically, let $R > 0$ be a constant guaranteed by Theorem 4.8 so that we get $Enc_{SC} : \{0,1\}^{Rb} \times \{0,1\}^{Rb} \to \{0,1\}^b$ is L_{SC}-weakly list decodable from radius $p + \alpha$ for $L_{SC} = 1$, and furthermore, Enc_{SC} is $2^{-2N^{\frac{1}{ad}}}$-pseudorandom for $\mathcal{C}'$.*

- *Let* $PRG : \{0,1\}^{(\log N)^{O(d')} \leq Rb} \to \{0,1\}^{N_{data}}$ *be an* $2^{-2N^{\frac{1}{ad}}}$*-PRG for* $\mathcal{C}'$ *from Theorem 3.6, and note that the seed length is smaller than* ℓ'_{ctrl}*.*

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) *specified in the figures has rate* $1 - H(p) - \epsilon$*, and is* $L = O(1)$*-list decodable with success probability* $1 - 2^{-N^{\frac{1}{ad}}}$ *against channels in* $\mathcal{C}$*. Furthermore,* Enc, Dec *are computable in time poly*(N) *for a fixed universal polynomial.*

6 Analyzing the construction

This section is devoted to proving Theorem 5.3.

The setup: Throughout the remainder of the section, we fix the following setup: Let $0 \leq p < \frac{1}{2}$ and $0 < \epsilon < \frac{1}{2} - p$ be constants. Let $\mathcal{C}, \mathcal{C}'$ be classes as required in Theorem 5.3. We use the choices and requirements made in Figure 1. More specifically, as in Figure 1, we assume that we are supplied with PRG and $(\mathrm{Enc}_{SC}, \mathrm{Dec}_{SC})$ that satisfy the requirements made in Figure 1. That is, that for some "required error" parameter $\nu \geq 2^{-\sqrt{N}}$ we have:

- A stochastic code $\mathrm{Enc}_{SC} : \{0,1\}^{2\log n_{\mathrm{ctrl}}} \times \{0,1\}^{\ell'_{\mathrm{SC}}} \to \{0,1\}^b$ that is ϵ_{SC}-pseudorandom for $\mathcal{C}'$ (for $\epsilon_{SC} = \frac{\nu}{10 \cdot n_{\mathrm{ctrl}}}$) and is L_{SC}-weakly list decodable from radius $p + \epsilon$, for a constant L_{SC}.
- An ϵ_{PRG}-PRG $\mathrm{PRG} : \{0,1\}^{\ell'_{\mathrm{ctrl}}} \to \{0,1\}^{N_{\mathrm{data}}}$ for $\mathcal{C}'$, for $\epsilon_{PRG} = \nu/10$.

Our goal in this section is to show that for every sufficiently large N, the encoding and decoding algorithms specified in Figures 2 and 3 satisfy the conclusion of Theorem 5.3. This setup is assumed throughout this section.

6.1 Milestones for correct decoding

Following Guruswami and Smith [GS10] we will analyze the construction in two steps: We first consider the case that the channel C is an additive channel, namely that $C(z) = z \oplus e$ for some fixed error vector e, and later extend to general channels that can choose e as a function of z.

We present the following abstraction of this method (which will be convenient for our purposes as we use several different classes of channels). 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.
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 computationally easy.

We will state a general theorem showing that if such milestones exist, then the correctness of the decoding holds even against channels that are not additive, as long as the construction is using pseudorandomness against a class $\mathcal{C}'$ that can simulate the channel and milestones. This is stated formally in the definition and theorem below (in which we allow milestones to be probabilistic).

Definition 6.1 (Milestones function). *Let* $A : \{0,1\}^{RN} \times \{0,1\}^{\ell'_{ctrl}} \times \{0,1\}^{\ell'_{ctrl}} \times \{0,1\}^{N} \times \{0,1\}^{N} \to \{0,1\}$ *be a function that receives as input: a message* $m \in \{0,1\}^{RN}$*, a sampler seed* $s_{samp} \in \{0,1\}^{\ell'_{ctrl}}$*, a permutation seed* $s_\pi \in \{0,1\}^{\ell'_{ctrl}}$*, an error vector* $e \in \{0,1\}^N$ *of relative hamming weight at most* p*, and a " random coins string"* $y \in \{0,1\}^N$*. We say that* A *is a* ***milestones function*** *(with respect to the classes* $\mathcal{C}, \mathcal{C}'$*) if it has all the following properties: (the probability space for the statements below is choosing the randomness of the encoder* $S = (S_{samp}, S_\pi, S_{PRG})$*,* $R = (R_1, \ldots, R_{n_{ctrl}})$ *and* Y *(the coins of* A*) uniformly and independently.)*

1. *For every* $m \in \{0,1\}^{RN}$*,* $s \in \{0,1\}^{\ell_{ctrl}}, r \in (\{0,1\}^{\ell'_{SC}})^{n_{ctrl}}$ *and* $e \in \{0,1\}^n$ *of relative hamming weight at most* p*,* $\Pr[A(m, s_{samp}, s_\pi, e, Y) = 1] \geq \frac{1}{2} \Rightarrow m \in Dec(Enc(m, s, r) \oplus e)$.
2. *For every* $m \in \{0,1\}^{RN}$ *and* $e \in \{0,1\}^n$ *of relative hamming weight at most* p*,* $\Pr[A(m, S_{samp}, S_\pi, e, Y) = 1] \geq 1 - \nu/10$.
3. *For every* $m, s_{samp}, s_\pi, y, C \in \mathcal{C}$*, every xored-restriction of the function* $D(z) = A(m, s_{samp}, s_\pi, E_C(z), y)$ *is in* $\mathcal{C}'$.

Lemma 6.2 (Milestones Lemma). *If there exist a milestones function with respect to* $\mathcal{C}, \mathcal{C}'$ *then*

$$\Pr[m \in Dec(C(Enc(m, S, R)))] \geq 1 - \nu$$

We defer the proof of the milestones lemma to Section 6.3. In the next section we explain how the milestones lemma implies Theorem 5.3.

6.2 Milestones Lemma implies Theorem 5.3

In this section we show that Lemma 6.2 implies Theorem 5.3. Our task is to define a milestone function that meets the three requirements in Definition 6.1. We start with the following definition.

Definition 6.3. *We say that a string* $e \in \{0,1\}^N$ *is* (λ, η)-***good*** *with respect to* $s_{samp} \in \{0,1\}^{\ell'_{ctrl}}$ *if for* $I = \{i_1, \ldots, i_{n_{ctrl}}\} = Samp(s_{samp})$*:*

$$\left| \left\{ j : \text{The Hamming weight of } e_{i_j} \text{ is at most } \lambda \cdot b \right\} \right| \geq \eta \cdot n_{ctrl}.$$

We will use slightly different milestone functions for different complexity measures (as we need the milestone function to be efficient for the corresponding complexity measure). It will be convenient to start by defining two milestone functions (a strong one, and a weak one). We will later show that more efficient milestone functions can be "sandwiched" between the two milestone functions. This will mean that correctness of the more efficient milestone functions will follow by analyzing the simpler versions.

Definition 6.4. *It will be convenient to denote the input to a milestone function by* (x, y) *where* $x = (m, s_{samp}, s_\pi, e)$ *and* y *is the "random coins", we define the following functions (which do not depend on* y*):*

Control milestone: *Let* $\mu = \epsilon^2/4$.

- $A^{weak}_{ctrl}(x, y) = 1$ *iff* e *is* $(p + \epsilon, \frac{\mu}{10})$*-good for* s_{samp}.
- $A^{strong}_{ctrl}(x, y) = 1$ *iff* e *is* $(p + \epsilon/4, (1 - \frac{1}{10}) \cdot \mu)$*-good for* s_{samp}.

Note that for every (x, y), $A^{strong}_{ctrl}(x, y) = 1 \Rightarrow A^{weak}_{ctrl}(x, y) = 1$.

Data milestone: *Let* $\pi_{s_\pi(\cdot)} = \pi(s_\pi, \cdot)$, $e_{data} = e^{Samp(s_{samp})}_{data}$, *and* $e^\pi = \pi_{s_\pi}(e_{data})$.

- $A^{weak}_{data}(x, y) = 1$ *iff* e^π *is* $(b', p + \alpha, \alpha)$*-balanced.*
- $A^{strong}_{data}(x, y) = 1$ *iff* e^π *is* $(b', p + \alpha/4, \alpha/4)$*-balanced.*

Note that for every (x, y), $A^{strong}_{data}(x, y) = 1 \Rightarrow A^{weak}_{data}(x, y) = 1$.

Combined milestones:

- $A^{weak}(x, y) = A^{weak}_{ctrl}(x, y) \wedge A^{weak}_{data}(x, y)$.
- $A^{strong}(x, y) = A^{strong}_{ctrl}(x, y) \wedge A^{strong}_{data}(x, y)$.

Note that for every (x, y), $A^{strong}(x, y) = 1 \Rightarrow A^{weak}(x, y) = 1$.

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

Lemma 6.5. *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 6.4.

Lemma 6.6. *The function* A^{strong} *satisfies the second property of a milestone function. (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 6.5.

Milestones for poly-size circuits. Both functions A^{weak}, A^{strong} satisfy the first two properties, and are obviously computable in polynomial time. This immediately gives that they satisfy the third and final property if $\mathcal{C}'$ is sufficiently stronger than $\mathcal{C}$ in the sense that it can run poly-time computations "on top of" computations in $\mathcal{C}$. This also immediately implies Theorem 5.3 for the case where A is allowed to run in some fixed polynomial time.

We would like to give tighter reductions in which the milestone function is computable in AC^0 or by a small space ROBP. We now explain how to achieve such milestone functions.

Milestone function for constant depth circuits. We would like to implement the milestone function A^{weak} (or A^{strong}) by a poly-size constant depth circuit. Note that the third property in Definition 6.1 considers the case that S_{samp}, S_π are fixed to some values s_{samp}, s_π, and the only live input is e. This means that the choice of permutation, and which blocks are control blocks is fixed (and can be hardwired as nonuniform advice) to the circuit. Furthermore, in the data milestone the inputs can be rearranged according to π_{s_π}, at no cost. Meaning that the circuit can compute

e^π from e at no cost. Thus, computing the milestone function reduces to several counting tasks on the number of ones in e and e^π.

It is known that the problem of counting the number of ones in an n bit input, cannot be solved by poly-size depth circuits. However, Ajtai [Ajt83] showed that for every $\eta > 0$, there is a polynomial size constant depth circuit that can produce a quantity that is the number of ones, up to an error of ηn. (In fact, the results of Ajtai are much stronger, and in particular allow subconstant η). This means that there is a circuit with constant depth and polynomial size $A'_{s_{\text{samp}},s_\pi}(e)$ such that for every m, y:

$$A^{strong}(m, s_{\text{samp}}, s_\pi, e, y) = 1 \Rightarrow A'_{s_{\text{samp}},s_\pi}(e) = 1 \Rightarrow A^{weak}(m, s_{\text{samp}}, s_\pi, e, y) = 1$$

This means that the milestone function $A^{middle}(x, y) = A'_{s_{\text{samp}},s_\pi}(e)$ satisfies the three properties of a milestone function proving Theorem 5.3 for the case of constant depth circuits.

Milestones for read once branching programs. As in the case of constant depth circuits, we need to implement the milestone function by an $O(\log n)$ space ROBP for fixed s_{samp}, s_π. Using the approach we used for constant depth circuits, this may seem easy at first glance, as ROBPs with space $O(\log n)$ can count up to $n^{O(1)}$ and this sufficed for the earlier implementation. Indeed, this reasoning applies to the control milestone, and the functions A^{strong}_{ctrl} and A^{weak}_{ctrl} can be easily implemented by an ROBP of space $O(\log n)$ (for fixed s_{samp}, s_π).

The functions A^{strong}_{data} and A^{weak}_{data} pose a problem. Unlike circuits, an ROBP is not allowed to reorder the input by a fixed permutation π_{s_π} prior to reading it. Thus, we cannot assume that online access to e, gives online access to e^π.

We do have that s_π is fixed, and can be hardwired to the ROBP. This means that when an ROBP reads the i'th bit of the input e, it can tell whether this bit belongs to a control block or a data block, and in the latter case, it can tell to which of the N_{data}/b' blocks of length b', does i belong to. (All these are operations that do not depend on e, and only depend on the fixed s_{samp}, s_π). The issue is that the order in which the ROBP reads the data bits is permuted, and does not respect their partitioning into blocks of length b'. This means that the ROBP cannot keep a single counter and use it for all blocks, and must maintain ℓ different counters, if it wants to count the number of ones in ℓ different blocks. The naive way to check if e^π is balanced, is to keep counters for all $\ell = N_{\text{data}}/b'$ blocks, and as b' is constant, this takes space $O(\ell) = O(N_{\text{data}}/b')$ which is way too much.

The solution is to use randomization. The milestone function is allowed to toss random coins (in the form of the input y). It will choose $\ell = O(\log N)$ uniform indices from $[N_{\text{data}}/b]$, and will only keep count of the number of ones in these blocks. (This can indeed be done in space $O(\log N)$). The milestone function will count the fraction of sampled blocks which have hamming weight larger than $p + \alpha/4$, and use this quantity ρ' as an approximation for the real quantity ρ (which is the fraction of blocks in e^π which have hamming weight larger than $p + \alpha/4$). By a Chernoff bound, with probability $1 - 2^{-\Omega(\alpha^2 \cdot \ell)} = 1 - N^{O(1)}$, we have that $|\rho - \rho'| \leq \alpha/100$. Therefore, the ROBP can safely output one if $\rho' \leq \alpha/2$, as this indeed implies that

$$A^{strong}_{data}(x, \cdot) = 1 \Rightarrow \Pr_Y[A^{middle}_{\text{data}}(x, Y) = 1] \geq 1 - 2^{-\Omega(\alpha^2 \ell)} \Rightarrow \Pr_Y[A^{middle}_{\text{data}}(x, Y) = 1] \geq \frac{1}{2} \Rightarrow A^{weak}_{\text{data}}(x, \cdot) = 1.$$

This gives that by Lemma 6.5, A^{middle} satisfies the first property of a milestone function. By Lemma 6.6, A^{middle} defined in this form, satisfies the second property of milestone functions, where

we suffer an additive loss of $2^{-\Omega(\alpha^2 \ell)}$ relative to what we can get for A^{strong}, because of the error induced by the Chernoff bound.

In Theorem 5.3, we are allowed to use space $O(\log N)$ for $\nu = 2^{-\Omega(\log N)}$, and as α is a constant, the Theorem follows.

6.3 Proof of Milestones Lemma

We prove the milestones lemma in two steps, described in the two sections below.

6.3.1 The hiding lemma

The following lemma states that for a function D that is slightly weaker than functions in $\mathcal{C}'$, an encoding of a message m is pseudorandom for D. (We will later consider the case where D is a composition of a channel and milestone functions).

Lemma 6.7 (Hiding Lemma). *Let D be a function such that every xored-restriction of D is in $\mathcal{C}'$. For every message $m \in \{0,1\}^{RN}$, sampler seed $s_{samp} \in \{0,1\}^{\ell'_{ctrl}}$ and permutation seed $s_\pi \in \{0,1\}^{\ell'_{ctrl}}$, let $V = Enc(m, s_\pi, s_{samp}, S_{PRG}, R_1, \cdots, R_{n_{ctrl}})$ 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 D, namely:*

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

Proof. We assume for contradiction that there exists D such that:

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

and note that $\epsilon_{PRG} + n_{\text{ctrl}} \cdot \epsilon_{SC} = \nu/5$. The lemma follows from the following claim.

Claim 6.8. *One of the following holds:*

- *There exists an xored-restriction C' of D such that, $|\Pr[C'(PRG(S_{PRG})) = 1] - \Pr[C'(U_{N_{data}}) = 1]| > \epsilon_{PRG}$.*
- *There exists $z' \in \{0,1\}^{2\log n_{ctrl}}$ and an xored restriction C' of D, such that $|\Pr[C'(Enc_{SC}(z', U_{\ell'_{SC}})) = 1] - \Pr[C'(U_b) = 1]| > \epsilon_{SC}$.*

Proof. (of claim) We partition V into $V = (V_{\text{data}}, V_{\text{ctrl}})^{\text{Samp}(s_{\text{samp}})}$ using definition 5.1. We have that D distinguishes $V = (V_{\text{data}}, V_{\text{ctrl}})$ from $U_N = (U_{data}, U_{ctrl})$ 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 distinguishes H from U_N with probability ϵ_{PRG},
- or, D distinguishes H from V with probability $n_{\text{ctrl}} \cdot \epsilon_{SC}$.

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 distinguishes $(U_{\text{data}}, v'_{\text{ctrl}})$ from $(V_{\text{data}}, v'_{\text{ctrl}})$ with probability ϵ_{PRG}. This gives that there exists an xored restriction of D that distinguishes U_{data} from V_{data} with probability ϵ_{PRG} and the first item of the claim holds.

In the second case, we have that m and s_π are fixed and therefore the string $y = \pi_{s_\pi}(\mathrm{Enc}_{\mathrm{balanced}}(m))$ used in the encoding algorithm is also fixed. The encoding algorithm computes the data part by xoring y with $PRG(S_{PRG})$ and therefore $V_{\mathrm{data}} = \mathrm{PRG}(S_{\mathrm{PRG}}) \oplus y$. By an averaging argument, there exists a fixing s'_{PRG} such that D distinguishes $((PRG(s'_{PRG}) \oplus y), U_{\mathrm{ctrl}})$ from $(((PRG(s'_{PRG}) \oplus y), V_{\mathrm{ctrl}})|S_{PRG} = s'_{PRG})$ with probability $n_{\mathrm{ctrl}} \cdot \epsilon_{SC}$. We have that there exists an xored restriction D' of D which distinguishes U_{ctrl} from $V'_{\mathrm{ctrl}} = (V_{\mathrm{ctrl}}|S_{PRG} = s'_{PRG})$.

Recall that the encoding procedure prepares the control part c_{ctrl} by preparing a string $z = \mathrm{Enc}_{LR}(s)$ and then the j'th control block is obtained by $\mathrm{Enc}_{SC}(z_j, r_j)$.

Having fixed $S_{PRG} = s'_{PRG}$ the only random variables that remain unfixed in V'_{ctrl} are $R_1, \ldots, R_{n_{\mathrm{ctrl}}}$. This means that there exists a fixed z such that $(V'_{\mathrm{ctrl}})_j = \mathrm{Enc}_{SC}(z_j, R_j)$ and in particular, the n_{ctrl} blocks are independent. We have that D' distinguishes V'_{ctrl} from U_{ctrl} with probability $n_{\mathrm{ctrl}} \cdot \epsilon_{SC}$, and by a standard hybrid argument, there exists an xored restriction C' of D' which distinguishes $(V'_{\mathrm{ctrl}})_j = \mathrm{Enc}_{SC}(z_j, R_j)$ from uniform with probability ϵ_{SC} and the second item follows. □

The lemma follows by the pseudorandomness properties of PRG and Enc_{SC}. □

6.3.2 Hiding lemma implies milestones lemma

We now show that the milestones lemma (Lemma 6.2) follows from the hiding lemma (Lemma 6.7). We are assuming that A is a milestone function with respect to $\mathcal{C}, \mathcal{C}'$ of Theorem 5.3. We need to show that for every message $m \in \{0,1\}^{RN}$, and every $C \in \mathcal{C}$,

$$\Pr[m \in \mathrm{Dec}(C(\mathrm{Enc}(m, S, R)))] \geq 1 - \nu$$

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

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

Claim 6.9. $\Pr[A(m, S_{samp}, S_\pi, E_C(Z), Y) = 1] < 1 - \nu/2$.

Proof. Let $B = \{(s,r)|m \notin \mathrm{Dec}(C(\mathrm{Enc}(m, s, r)))\}$ be the set of pairs on which C causes a decoding error. We have that $\Pr[(S,R) \in B] \geq \nu$.

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). Let $A' = A(m, S_{\mathrm{samp}}, S_\pi, E_C(Z), Y)$ be the random variable of the output of function A in the probability space where S, R, Y are chosen uniformly.

$$\Pr[A' = 0] \geq \Pr[A' = 0|(S,R) \in B] \cdot \Pr[(S,R) \in B] > \nu/2$$

It follows that

$$\Pr[A(m, S_{\mathrm{samp}}, S_\pi, E_C(Z), Y) = 1] = \Pr[A' = 1] = 1 - [A' = 0] < 1 - \nu/2.$$

□

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,

$$\Pr[A(m, S_{\text{samp}}, S_\pi, e, Y) = 1] \geq 1 - \nu/10.$$

As Z_U is independent of (S_{samp}, S_π) this holds also for an error vector of the form $E_C(Z_U)$. Namely,

$$\Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z_U), Y) = 1] \geq 1 - \nu/10.$$

This means that:

$$\begin{aligned}\Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z_U), Y) = 1] - \Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z), Y) = 1] \\ > (1 - \nu/10) - (1 - \nu/2) \geq \nu/4.\end{aligned}$$

By averaging, there exist fixed values s'_{samp}, s'_π and y' such that if we consider the event $W = \{S_{\text{samp}} = s'_{\text{samp}}, S_\pi = s'_\pi, Y = y'\}$.

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

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

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

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 6.7). By the third property of milestones functions, we have that every xored restriction of the function $D(z) = A(m, s'_{\text{samp}}, s'_\pi, E_C(z), y')$ is in $\mathcal{C}'$. Therefore, the function D that we obtained gives a contradiction to the hiding lemma.

6.4 Proof of Lemma 6.5

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

Claim 6.10. *For every $m, s = (s_{samp}, s_\pi, s_{PRG}), r, e$ and y, let $c = Enc(m, s, r)$ and $c' = c \oplus e$. If $A^{weak}_{ctrl}(m, s_{samp}, s_\pi, e, y) = 1$ then $s \in List_{ctrl}$. Where $List_{ctrl}$ is the list obtained in the decoding algorithm described in Figure 3.*

Proof. Recall that $\text{List}_{\text{ctrl}} = Dec_{LR}(List_{SC})$, $List_{SC} = \cup_{i\in[n]} List_i$ and $\text{List}_i = \text{Dec}_{SC}(c'_i)$ (here c'_i is the i'th block of c'). By Definition 6.1, $A^{weak}_{\text{ctrl}}(x) = 1$ iff e is $(p+\epsilon, \frac{\mu}{10})$-good for s_{samp}. Let e_i denote the error vector restricted to the i'th block. By the properties of Enc_{SC}, if the hamming weight of e_i is less than $(p+\epsilon)\cdot b$ then $c_i \in \text{List}_i$. We have that e is $(p+\epsilon, \frac{\mu}{10})$-good for s_{samp}, and this means that for at least $\frac{\mu}{10} \cdot n_{\text{ctrl}} = \frac{\epsilon^2 \cdot n_{\text{ctrl}}}{40}$ of the n_{ctrl} control blocks $i \in I = \text{Samp}(S_{\text{samp}})$, $c_i \in \text{List}_{SC} = \cup_{i\in[n]} List_i$. Thus, we indeed have that $\Pr_{i\leftarrow[n_{\text{ctrl}}]}[\text{Enc}_{LR}(s)_i \in List_{SC}] \geq \frac{\mu}{10} > \frac{\epsilon^2}{100}$ for a set List_{SC} of size $n \cdot L_{SC}$. By the list recoverability of Enc_{LR} we get that $s \in \text{List}_{\text{ctrl}}$ meaning that the control information was successfully recovered as desired.

□

Claim 6.11. *For every $m, s = (s_{samp}, s_\pi, s_{PRG}), r, e$ and y, let $c = Enc(m, s, r)$ and $c' = c \oplus e$. If $A^{weak}_{data}(m, s_{samp}, s_\pi, e, y) = 1$ and $s \in List_{ctrl}$ (meaning that s was recovered correctly by the first step of decoding) then $m \in Dec(c')$.*

Proof. We have that $s \in \mathrm{List}_{\mathrm{ctrl}}$, 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 $A^{weak}_{\mathrm{data}}(m, s_{\mathrm{samp}}, s_\pi, e, y) = 1$ implies that e^π is $(b', p + \alpha, \alpha)$-balanced. Note that e^π is the error vector used on the balanced code. By the guarantee on $\mathrm{Dec}_{\mathrm{balanced}}$ this gives that $m \in \mathrm{List}_s = \mathrm{Dec}_{\mathrm{balanced}}(y')$, since the correct control is in $\mathrm{List}_{\mathrm{ctrl}}$ then $m \in \mathrm{Dec}(c') = \bigcup_{s \in \mathrm{List}_{\mathrm{ctrl}}} \mathrm{List}_s$ as desired. □

The lemma follows from the combination of both claims.

6.5 Proof of Lemma 6.6

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 p.

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 6.12. *For every m, e of hamming weight at most pn, y, and s_π,*

$$\Pr[A^{strong}_{ctrl}(m, S_{samp}, s_\pi, e, y) = 1] \geq 1 - 2^{-N^{0.6}}.$$

Proof. For a given error vector e we define $T_e = \left\{i : \text{The ith block has a weight at most } (p + \frac{\epsilon}{4}) \cdot b\right\}$. 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,

$$\Pr_{(z_1,\ldots,z_{n_{\mathrm{ctrl}}}) \leftarrow \mathrm{Samp}(U_{\ell'_{\mathrm{ctrl}}})} [|\frac{1}{n_{\mathrm{ctrl}}}|\{i : z_i \in T_e\}| - \frac{|T_e|}{n}| > \frac{\epsilon^2}{100}] \leq 2^{-N^{0.6}}.$$

Thus, if we choose S_{samp} uniformly and independently we get that with probability $1 - 2^{-N^{0.6}}$, the number of control blocks that are good (have error less than $p + \frac{\epsilon}{4}$) is at least $(\frac{\epsilon}{4} - \frac{\epsilon^2}{100})n_{\mathrm{ctrl}} > (\frac{9}{10} \cdot \epsilon^2/4)n_{\mathrm{ctrl}} = ((1 - \frac{1}{10}) \cdot \mu)n_{\mathrm{ctrl}}$. This means that the error vector e is $(p + \epsilon/4, (1 - \frac{1}{10}) \cdot \mu)$-good with probability $1 - 2^{-N^{0.6}}$ and the claim holds. □

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

Claim 6.13. *For every m, e of hamming weight at most pN, y, and s_π,*

$$\Pr_{s_{samp} \leftarrow U_{\ell'_{ctrl}}} [weight(e^{Samp(s_{samp})}_{data}) \geq N_{data} \cdot (p + \frac{\alpha}{100})] \leq 2^{-N^{0.6}}$$

(here, weight is hamming weight).

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

$$\Pr_{(z_1,\ldots,z_{n_{\text{ctrl}}})\leftarrow \text{Samp}(U_{\ell'_{\text{ctrl}}})}[|\frac{1}{n_{\text{ctrl}}}\sum_{i\in[n_{\text{ctrl}}]} f(z_i) - p| > \frac{\alpha}{100}] \leq 2^{-N^{0.6}}.$$

Thus with probability $1 - 2^{-N^{0.6}}$ the number of errors induced to the control blocks is at least $N_{\text{ctrl}}(p - \frac{\alpha}{100})$, which implies that the number of error induced to the data is less than $N_{\text{data}}(p + \frac{\alpha}{100})$, and the claim follows. □

We will now show that permuting the data part e, produces a balanced error vector with high probability. Let s_{samp} be a sampler seed that is good with respect to the two previous claims. A $1 - 2 \cdot 2^{-N^{0.6}}$ fraction of sampler seeds, satisfy these properties. By Claim 6.13, we can assume that the relative hamming weight of $e_{\text{data}}^{s_{\text{samp}}}$ is at most $p + \alpha/100$. We will denote $e_{\text{data}} = e_{\text{data}}^{s_{\text{samp}}}$ in order to avoid clutter. The lemma will follow from the following claim.

Claim 6.14. $\Pr[\pi(S_\pi, e_{data})$ *is* $(b', p + \alpha/4, \alpha/4)$*-balanced error*$] > 1 - e^{-\Omega(N^{0.55})}$.

This is because, together the three claims above give that with probability $1 - 2^{-N^{0.51}}$ all good events happen, and $A^{strong}(x, y) = 1$. In the remainder of this section we prove Claim 6.14.

Let $N' = N_{\text{data}}/b'$ be the number of b' length blocks. We now define random variables $D_1, \ldots, D_{N'}$ as follows.

$$D_i = \begin{cases} 1 & \text{The i'th block of } \pi(S_\pi, e_{\text{data}}) \text{ has weight more than } (p + \frac{\alpha}{4}) \cdot b' \\ 0 & \text{otherwise} \end{cases}$$

Claim 6.14 can now be seen as a claim that the sum of the D_i's is small with high probability. We will use a Chernoff style bound, due to Schmidt, Siegel and Srinivasan [SSS95] in order to bound the probability of deviation.

Lemma 6.15. *[SSS95] Suppose* $X_1, ..., X_\ell$ *are binary random variables, such that for every set of distinct* k *indices* $i_1, \cdots, i_k \in [\ell]$, $\Pr[X_{i_1} = \ldots = X_{i_k} = 1] \leq \mu^k$. *If* $0 < \delta \leq 1$ *and* $k \leq \frac{\delta \cdot \mu \cdot \ell}{2}$ *then*

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

We plan to use Lemma 6.15 on the random variables $D_1, \ldots, D_{N'}$ for this purpose, we need to analyze the probability that tuples of D_i's all evaluate to one. In order to achieve this, we will first show that:

Claim 6.16. *For every* $v < N^{5.5}$ *and every distinct* $i_1, \ldots, i_v \in [N']$, *and additional* $i \in [N']$

$$\Pr[D_i = 1 | D_{i_1} = \ldots = D_{i_v} = 1] \leq \alpha/10$$

We observe that Claim 6.16 implies Claim 6.14 by Lemma 6.15. This is because Claim 6.16 implies that for $v = N^{5.5}$, and every distinct $i_1, \ldots, i_v \in [N']$,

$$\Pr[D_{i_1} = \ldots = D_{i_v} = 1] \leq (\alpha/10)^v.$$

We can now use Lemma 6.15 with $k = N^{5.5}$, $\delta = 1$ and $\mu = \alpha/10$ to get that:

$$Pr[\sum_{j=1}^{N'} D_j \geq \frac{\alpha \cdot N'}{5}] \leq e^{-\Omega(N^{5.5})}$$

In order to prove Claim 6.16 we prove the following claim, for which we introduce the following notation: We use e^{s_π} to denote $\pi_{s_\pi}(e_{\text{data}})$. We use $e^{s_\pi}[i]$ to denote the i'th block of e^{s_π} (where blocks are of length b'). We use $e^{s_\pi}[i,j]$ to denote the j'th bit in the i'th block of e^{s_π}.

Claim 6.17. *Let $v < N^{0.55}$, let $i_1, \ldots, i_v \in [N']$ be distinct blocks, let $i \in [N']$ be an additional block, and let $j_1, \ldots, j_k \in [b']$. Let $a_1, \ldots, a_v \in \{0,1\}^{b'}$ be strings such that the relative hamming weight of each a_i is at least $p + \alpha/100$. Let $E = \cap_{m \in [v]} \left\{e^{S_\pi}[i_m] = a_m\right\}$. It follows that:*

$$\Pr[\cap_{\ell \in [k]} \left\{e^{S_\pi}[i, j_\ell] = 1\right\} | E] \leq (p + \alpha/50)^k$$

Proof.

$$\Pr[\cap_{\ell \in [k]} \left\{e^{S_\pi}[i, j_\ell] = 1\right\} | E] = \frac{\Pr[\cap_{\ell \in [k]} \left\{e^{S_\pi}[i, j_\ell] = 1\right\} \cap E]}{\Pr[E]}$$

Let us first imagine that π is an $(0,t)$-wise independent permutation. In this case, the denominator is some quantity $\beta \geq 1/N_{\text{data}}^v \geq 1/N^{N^{0.55}} \geq 1/2^{N^{0.56}}$ and the enumerator is at most $\beta \cdot (p + \alpha/100)^k$. This is because conditioned on the v values, the fraction of ones that is "still available" in e_{data} has not increased, and is still at most $p + \alpha/100$. It follows that the actual quantity is at most

$$\frac{\beta \cdot (p + \alpha/100)^k + 2^{-N^{0.6}}}{\beta - 2^{-N^{0.6}}} = \frac{(p + \alpha/100)^k + 2^{-N^{0.6}}/\beta}{1 - 2^{-N^{0.6}}/\beta} \leq (p + \alpha/50)^k$$

where the last inequality follows for sufficiently large N because p, α and $k \leq b'$ are constants, and for every two constants $A < A'$, $\frac{A + o(1)}{1 - o(1)} \leq A'$. □

We now show that Claim 6.16 follows directly from Claim 6.17, using Lemma 6.15.

Proof. (of Claim 6.16) We use Lemma 6.15 on the random variables $Y_1, \ldots, Y_{b'}$ defined by:

$$Y_w = \begin{cases} 1 & e^{S_\pi}[i, w] = 1 \\ 0 & \text{otherwise} \end{cases}$$

By Claim 6.17 we have that for every $0 \leq v < N^{5.5}$, and for every k-tuple of indices $j_1, \ldots, j_k \in [b']$ in the i'th block,

$$\Pr[Y_{j_1} = \ldots = Y_{j_k} = 1 | D_{i_1} = \ldots = D_{i_v} = 1] \leq (p + \alpha/50)^k.$$

Applying Lemma 6.15, with $\delta = \alpha/10$, $k = \alpha^2 \cdot b'/2$, $\mu = p + \alpha/50$, and noting that $(1+\delta) \cdot \mu \leq p + \alpha/4$ we have that:

$$Pr[\sum_{j=1}^{b'} Y_j \geq (p + \alpha/4) \cdot b' | D_{i_1} = \ldots = D_{i_v} = 1] \leq e^{-\Omega(\alpha^3 \cdot b')} \leq \alpha/10,$$

where the last inequality follows as we are allowed to choose b' to be a sufficiently large constant as a function of α, and the claim follows. □

7 Proof of Theorem 3.10

In this section we prove Theorem 3.10. The high level idea is that concatenated codes easily give codes for balanced errors. A similar argument also appears in [Smi07], for the case of codes against errors that are "t-wise independent".

Codes with the property required in Theorem 3.10 can be constructed by concatenating:

- An explicit outer code $C_{out} : \{0,1\}^k \to (\{0,1\}^{n_{in}})^{n_{out}}$ that is $(1-\gamma, L_{in}, L)$-list recoverable from a collection, that has rate at least $1 - \epsilon/3$, and in which n_{in}, L_{in}, L are constants and $L = \text{poly}(1/\epsilon) \cdot L_{in}$.
- An inner code $C_{in} : \{0,1\}^{n_{in}} \to \{0,1\}^b$ that is L_{in}-list-decodable from $p \cdot b$ errors and has rate at least $1 - H(p) - \epsilon/3$.

Note that this indeed gives a code with the desired properties: The inner code can be list-decodable in constant time by brute force. Furthermore, for balanced error, list-decoding succeeds on $1-\gamma$ of the n_{out} blocks, giving that the list-recovering algorithm of the outer code, is set up to output a list containing the original message.

For every constant $\epsilon > 0$ if we choose sufficiently large constants n_{in}, b and $L_{in} = \text{poly}(1/\epsilon)$ then inner codes with the required property exist by a standard probabilistic argument, and as C_{in} is of constant size, we can find such codes by brute force search.
The outer code can be constructed by concatenating:

- An explicit code $C_1 : \{0,1\}^k \to (\{0,1\}^{\log n_1})^{n_1}$ that is $(1-\gamma^2, L_1, L_2 = L)$-list recoverable from a collection, and has rate at least $1 - \epsilon/9$. We need that $L = \text{poly}(1/\epsilon) \cdot L_1$.
- An inner code $C_2 : \{0,1\}^{\log n_1} \to (\{0,1\}^{n_{in}})^{n_2}$ that is $(1-\gamma^2, L_{in}, L_1)$-list recoverable from a collection, and has rate at least $1 - \epsilon/9$, and in which $n_{in}, L_{in}, L_1 = \text{poly}(1/\epsilon)$ are constants.

This gives $n_{out} = n_1 \cdot n_2$, and the correctness follows as concatenation of list-recoverable codes gives a list recoverable codes. Specifically: Given a collection of $n_{out} = n_1 \cdot n_2$ sets (indexed by $(i_1, i_2) \in [n_1] \times [n_2]$), $T_{(i_1,i_2)} \subseteq \{0,1\}^{in}$ of size L_{in}, we need to list recover a list of size at most L, containing all $m \in \{0,1\}^k$ such that

$$\Pr_{(i_1,i_2) \leftarrow [n_1]\times[n_2]}[\text{Enc}_{C_{out}}(m)_{(i_1,i_2)} \in T] \geq 1 - \gamma^2.$$

By averaging, for every such m, we have that for a $1-\gamma$ fraction of $i_1 \in [n_1]$,

$$\Pr_{i_2 \leftarrow [n_2]}[\text{Enc}_{C_2}(\text{Enc}_{C_1}(m)_{i_1}) \in T] \geq 1 - \gamma.$$

and so performing two steps of list-recovering indeed recovers the original message.

The outer code C_1 can be taken to be a Reed-Solomon code, and by [Sud97, GS99], we get these parameters if $\epsilon \leq O(\gamma^2)$ for $L_2 = \text{poly}(1/\epsilon) \cdot L_1$. We now turn our attention to the inner code C_2. We will use the probabilistic method to show the existence of a good code, and such code can be later found by exhaustive search.

Claim 7.1. *There exists a constant $c > 1$, such that for every sufficiently small constants $\epsilon > 0$ and $\gamma > 0$ such that $\epsilon \leq \gamma^c$ and every constant L_{in}, there exist constants $L_1 = L_{in} \cdot \text{poly}(1/\epsilon)$ and $n_{in} \geq \frac{\log L_{in}}{\gamma}$, such that for every sufficiently large k_2, there is a code $C_2 : \{0,1\}^{k_2} \to (\{0,1\}^{n_{in}})^{n_2}$ that is $(1-\gamma^2, L_{in}, L_1)$-list recoverable from a collection and has rate $1 - \epsilon/9$.*

Proof. We consider a uniformly chosen C_2. For every subset $S \subseteq \{0,1\}^{k_2}$ of size L_1+1, and every collection T of sets $T_1, \ldots, T_2 \subseteq \{0,1\}^{n_{in}}$ of size L_{in} let $B^{S,T}$ be the event that for every $x \in S$, for a $1-\gamma^2$ fraction of $i \in [n_2]$, $\mathrm{Enc}_{C_2}(x)_i \in T_i$. Our goal is to do a union bound over all of these events. We will choose n_{in} to be sufficiently large so that $L_{in} \leq 2^{\gamma n_{in}}$. Let $N_{in} = 2^{n_{in}}$ and let $\alpha = L_{in}/N_{in}$ so that $\log(1/\alpha) = (1-\gamma) \cdot n_{in}$. Note that for fixed x and a collection T, we can use a Chernoff bound[13], to show that the probability that a $1-\gamma^2$ fraction of $i \in [n_2]$, $C_2(x)_i \in T_i$, is at most

$$2^{-(1-\gamma^2)\cdot n_2 \cdot \log \frac{1-\gamma^2}{e\cdot\alpha}} \leq 2^{-(1-\gamma^2)\cdot n_2 \cdot \log \frac{10}{\alpha}}$$

where the last inequality follows for sufficiently small γ. It follows that for every S, T:

$$\Pr[B^{S,T}] \leq 2^{-(L+1)\cdot(1-\gamma^2)\cdot n_2 \cdot \log \frac{10}{\alpha}}$$

The number of choices for S, T is bounded by:

$$\binom{2^{k_2}}{L_1+1} \cdot \binom{N_{in}}{L_{in}}^{n_2} \leq 2^{(L_1+1)\cdot k_2} \cdot \left(\frac{e \cdot N_{in}}{L_{in}}\right)^{n_2 \cdot L_{in}} \leq 2^{(L_1+1)\cdot k_2} \cdot 2^{n_2 \cdot L_{in} \cdot \log \frac{e}{\alpha}}.$$

Thus, we can do a union bound if:

$$k_2 < (1-\gamma)\cdot(1-\gamma^2)\cdot n_2 \cdot \log\frac{10}{\alpha} = (1-\gamma^2)\cdot(1-\gamma)^2 \cdot n_2 \cdot n_{in},$$

and also,

$$L_{in} \cdot \log\frac{e}{\alpha} < \gamma \cdot (L+1) \cdot (1-\gamma^2) \cdot \log\frac{10}{\alpha}.$$

The first inequality follows because we are allowed to choose $k_2 = (1-\epsilon/9)\cdot n_2 \cdot n_{in}$, and ϵ is was chosen to be sufficiently smaller than γ. The second inequality follows as we are allowed to choose $L_1 = L_{in} \cdot \mathrm{poly}(1/\epsilon)$, and $\gamma \geq \epsilon$. □

The inner code C_2 is over an alphabet of logarithmic size in k_{out}, and can be found (and decoded) by brute force search in time polynomial in k_{out}.

Acknowledgement

We are grateful to Swastik Kopparty for pointing us to the Algebraic Geometric codes of Garcia and Stichtenoth, and in particular for pointing us to their description in [Shp09].

References

[Ajt83] Miklós Ajtai. Σ^1_1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.

[Bra10] M. Braverman. Polylogarithmic independence fools ac^0 circuits. *J. ACM*, 57(5), 2010.

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

[13] The Chernoff bound we use is that if $X_1, \ldots, X_n$ are independent indicator random variables, and the expectation of their sum X is μn, then for $v > 10$, $\Pr[X \geq v \cdot \mu \cdot n] \leq 2^{-v\cdot\mu\cdot n\cdot\ln(v/e)}$.

[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.

[GS10] Venkatesan Guruswami and Adam D. Smith. Codes for computationally simple channels: Explicit constructions with optimal rate. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 723–732, 2010.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 356–364, 1994.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

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

[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.

[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.

[Nis91] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS: Journal of Computer and System Sciences*, 49, 1994.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[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.

[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.

[Tal14] Avishay Tal. Tight bounds on the fourier spectrum of ac^0. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:174, 2014.

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC*, pages 242–247, 2013.

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

ECCC ISSN 1433-8092

http://eccc.hpi-web.de