Codes for Computationally Simple Channels: Explicit Constructions with Optimal Rate

(Extended Abstract)

Venkatesan Guruswami
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA

Adam Smith
Computer Science & Engineering Department
Pennsylvania State University
University Park, USA

***Abstract*—In this paper, we consider coding schemes for *computationally bounded* channels, which can introduce an arbitrary set of errors as long as (a) the fraction of errors is bounded with high probability by a parameter p and (b) the process which adds the errors can be described by a sufficiently "simple" circuit. Codes for such channel models are attractive since, like codes for standard adversarial errors, they can handle channels whose true behavior is *unknown* or *varying* over time.**

For three classes of channels, we provide explicit, efficiently encodable/decodable codes of optimal rate where only *in*efficiently decodable codes were previously known. In each case, we provide one encoder/decoder that works for *every* channel in the class.

***Unique decoding for additive errors:* We give the first construction of a poly-time encodable/decodable code for *additive* (a.k.a. *oblivious*) channels that achieve the Shannon capacity $1 - H(p)$.**

***List-decoding for online log-space channels:* We give an efficient code with optimal rate (arbitrarily close to $1 - H(p)$) that recovers a short list containing the correct message with high probability for channels which read and modify the transmitted codeword as a stream, using at most $O(\log N)$ bits of workspace on transmissions of N bits.**

***List-decoding for poly-time channels:* For any constant c we give a similar list-decoding result for channels describable by circuits of size at most N^c, assuming the existence of pseudorandom generators.**

I. Introduction

For the binary symmetric channel which flips each transmitted bit independently with probability $p < 1/2$, the optimal rate of reliable transmission is known to be the Shannon capacity $1-H(p)$, where $H(\cdot)$ is the binary entropy function [24]. Moreover, concatenated codes approach this capacity and are efficiently decodable [8]. In contrast, for adversarial channels that can corrupt up to a fraction p of symbols in an *arbitrary* manner, the optimal rate is unknown, though it is known that the rate has to be much smaller than the Shannon capacity. In particular, for $p > 1/4$, the achievable rate over an adversarial channel is zero, while $1 - H(p)$ remains positive. Determining the best asymptotic rate for error fraction p (equivalently, minimum relative distance $2p$) remains an important open question in combinatorial coding theory.

Codes that tolerate adversarial errors are attractive because they can transmit reliably over a large range of channels whose true behavior is unknown or varies over time. In contrast, codes tailored to a specific channel model tend to fail when the model changes. For example, concatenated codes, which can transmit efficiently and reliably at the Shannon capacity with i.i.d. errors, fail miserably in the presence of *burst* errors that occur in long runs.

In this paper, we consider several intermediate models of uncertain channels. Specifically, we consider *computationally bounded* channels, which can introduce an arbitrary set of errors as long as (a) the total fraction of errors is bounded by p with high probability and (b) the process which adds the errors can be described by a sufficiently "simple" circuit. The idea behind these models is that natural processes may be mercurial, but are not computationally intensive. These models are powerful enough to capture natural settings like *i.i.d.* and burst errors, but weak enough to allow efficient communication arbitrarily close to the Shannon capacity. The models we study, or close variants, have been considered previously—see Section II for a discussion of related work. The computational perspective we espouse is inspired by the works of Lipton [20] and Micali *et al.* [21].

For three classes of channels, we provide efficiently encodable and decodable codes of optimal rate (arbitrar-

[1]A full version of this paper is available online [13].

[2]V.G. is supported by a Packard Fellowship and NSF grants CCF-0953155 and CCF-0963975. A.S. is supported by NSF grants TF-0747294 and TF-0729171.

ily close to $1-H(p)$) where only *in*efficiently decodable codes were previously known. In each case, we provide one encoder/decoder that works for *every* channel in the class. In particular, our results apply even when the channel's behavior depends on the code.

We first describe the models and our results briefly (Section I-A). In Section II, we describe related lines of work aimed at handling (partly) adversarial errors with rates near Shannon capacity. Our results are stated formally in Section III.

A. Our results

The encoders we construct are *stochastic* (that is, randomized). Probabilities are taken over the (local, unknown to the decoder) coins of the encoder and the choices of the channel; messages may be chosen adversarially and known to the channel. Our results assume *no* setup or shared randomness between the encoder and decoder.

Unique decoding for additive channels: We give the first explicit construction of stochastic codes with polynomial-time encoding/decoding algorithms that approach the Shannon capacity $1 - H(p)$ for *additive* (a.k.a. *oblivious*) channels. These are channels which add an arbitrary error vector $e \in \{0, 1\}^N$ of Hamming weight at most pN to the transmitted codeword (of length N). The error vector may depend on the code but, crucially, not on the encoder's local random coins. Additive errors capture binary symmetric errors as well as certain models of correlated errors, like burst errors. For a deterministic encoder, the additive error model is equivalent to the usual adversarial error model. A randomized encoder is thus necessary to achieve the Shannon capacity.

We also provide a novel, simple proof that (inefficient) capacity-achieving codes *exist* for additive channels. We do so by combining linear list-decodable codes with rate approaching $1-H(p)$ (known to exist, but not known to be efficiently decodable) with a special type of authentication scheme. Previous existential proofs relied on complex random coding arguments [2], [18]; see the discussion of related work below.

List decoding for space-bounded channels: The additive errors model postulates that the error vector has to be picked obliviously, before seeing the codeword. To model more complex processes, we consider a channel that processes the codeword as a stream, deciding as it goes which positions to corrupt. The channel's only limitation is a bound $S(N)$ on the amount of work space it can use (as a function of the block length N). More precisely, in order to allow *nonuniform* dependency on the code, we model the channel as a width-$2^{S(N)}$ branching program that outputs one bit for every input bit that it reads. Even for constant space S, this model captures a wide range of channels considered in coding theory, including additive channels, discrete channels with finite memory, echo, bounded delay and *arbitrarily varying channels* (see the discussion of related work below for definitions). As above, we assume that, with high probability, the channel introduces at most pN errors.

First, we show that reliable *unique* decoding with positive rate is impossible even if one only wants to tolerate an arbitrary *memoryless* channel, when $p > 1/4$. The idea is that even a memoryless adversary can make the transmitted codeword difficult to distinguish from a different, random codeword. The proof relies on the requirement that a single code must work for all channels, since the "hard" channel depends on the code.

Thus, to communicate at a rate close to $1 - H(p)$ for all p, we consider the relaxation to *list-decoding*: the decoder is allowed to output a small list of messages, one of which is correct. List-decodable codes with rate approaching $1 - H(p)$ are known to exist even for adversarial errors [29], [7]. However, constructing efficient (*i.e.*, polynomial-time encodable and decodable) codes for list decoding with near-optimal rate is a major open problem.

Our main contribution for space-bounded channels is a construction of polynomial-time list-decodable codes with optimal rate for channels whose space bound is logarithmic in the block length N. Note that the decoder need *not* return all words within a distance pN of the received word (as is the case for the standard "combinatorial" meaning of list decoding for deterministic codes), but it *must* return the correct message as one of the candidates with high probability. From a communication viewpoint, this notion of list-decoding is natural for stochastic codes. In fact, it is exactly the notion that is needed in constructions which "sieve" the list, such as [10], [21]; see related work in Section II.

List decoding for polynomial time channels: More generally, one may consider channels whose behavior on N-bit inputs is described by a circuit of size $T(N)$. Logarithmic space channels, in particular, can be realized by polynomial-size circuits. In fact, we do not know of any channel models considered in the information theory literature, other than purely adversarial channels, which require more than linear time to implement. Our construction of list-decodable codes for logarithmic-space channels can be extended to handle channels with a given polynomial time bound $T(n) =$

N^c, for any fixed $c > 1$, under an additional assumption, namely, the existence of pseudorandom generators of constant stretch that output N pseudorandom bits and fool circuits of size N^c. Such generators exist, for example, if there are functions in E which have no subexponential-size circuits [23], [15], or if one-way functions exist [28], [14].

II. Background and Related Previous Work

There are several lines of work aimed at handling adversarial, or partly adversarial, errors with rates near the Shannon capacity. We survey them briefly here and highlight the relationship to our results.

List decoding: List decoding was introduced in the late 1950s [6], [27] and has witnessed a lot of recent algorithmic work (cf. the survey [11]). Under list decoding, the decoder outputs a small list of messages that must include the correct message. Random coding arguments assert the existence of binary codes of rate $1 - H(p) - \varepsilon$ for which error-correction against adversarial errors is possible in this model when the decoder is allowed to output a list of size $O(1/\varepsilon)$ [7], [29], [12]. If a small amount of auxiliary information can be communicated on a noiseless side channel, then it becomes possible to pick the correct element from the list with high probability [10]. The explicit construction of binary list-decodable codes with rate close to $1 - H(p)$, however, remains a major open question. We provide such codes for the special case of corruptions introduced by space- or time-bounded channels.

Adding Setup—Shared Randomness: Another relaxation is to allow randomized coding strategies where the sender and receiver share "secret" randomness, *hidden from the channel*, which is used to pick a particular, deterministic code at random from a family of codes. Such randomized strategies were called *private codes* in [17]). Using this secret shared randomness, one can achieve the capacity $1-H(p)$ against adversarial errors (for example, by randomly permuting the symbols and adding a random offset [20], [17]). Using explicit codes achieving capacity on the BSC_p [8], one can even get such randomized codes of rate approaching $1 - H(p)$ explicitly (although getting an explicit construction with $o(n)$ randomness remains an open problem [25]). A related notion of setup is the *public key* model of Micali *et al.* [21], in which the sender generates a public key that is known to the receiver and possibly to the channel. This model only makes sense for computationally bounded channels, discussed below.

AVCs: Oblivious, nonuniform errors: A different approach to modeling uncertain channels is embodied by the rich literature on *arbitrarily varying channels* (AVCs), surveyed in [19]. Despite being extensively investigated in the information theory literature, AVCs have not received much algorithmic attention.

An AVC is specified by a finite state space $\mathcal{S}$ and a family of memoryless channels $\{W_s : s \in \mathcal{S}\}$. The channel's behavior is governed by its state, which is allowed to vary arbitrarily. The AVC's behavior in a particular execution is specified by a vector $\vec{s} = (s_1, ..., s_N) \in \mathcal{S}^N$: the channel applies the operation W_{s_i} to the ith bit of the codeword. A code for the AVC is required to transmit reliably with high probability for every sequence $\vec{s}$, possibly subject to some state constraint. Thus AVCs model uncertainty via the *nonuniform* choice of the state vector $\vec{s} \in \mathcal{S}^N$. However — and this is the one of the key differences that makes the bounded space model more powerful — the choice of vector $\vec{s}$ in an AVC is *oblivious* to the codeword; that is, the channel cannot look at the codeword to decide the state sequence.

The additive errors channel we consider *is* captured by the AVC framework. Indeed, consider the simple AVC where $\mathcal{S} = \{0, 1\}$ and when in state s, the channel adds s mod 2 to the input bit. With the state constraint $\sum_{i=1}^{N} s_i \leqslant pN$ on the state sequence $(s_1, s_2, \ldots, s_N)$ of the AVC, this models additive errors, where an *arbitrary* error vector e with at most p fraction of 1's is added to the codeword by the channel, but e is chosen *obliviously* of the codeword.

Csiszár and Narayan determined the capacity of AVCs with state constraints [3], [4]. In particular, for the additive case, they showed that random codes can achieve rate approaching $1 - H(p)$ while correcting any specific error pattern e of weight pN with high probability.[1] Note that codes providing this guarantee *cannot* be linear, since the bad error vectors for all codewords are the same in a linear code. Langberg [18] gave another proof of the above claim, based on a different random coding argument.

As outlined above, we provide two results for this model. First, we give a new and simpler existential proof. More importantly, we provide the first explicit

[1]The AVC literature usually discusses the "average error criterion", in which the code is deterministic but the message is assumed to be uniformly random and unknown to the channel. We prefer the "stochastic encoding" model, in which we consider the worst-case message, but allow the encoder some local random coins. This is a strict strengthening of the model as long as the decoder recovers the random coins r along with message m. The results of [3], [18] also apply to this stronger model.

constructions of codes for this model which achieve the optimal rate $1 - H(p)$.

Polynomial-time bounded channels: In a different vein, Lipton [20] considered channels whose behavior can be described by a polynomial-time algorithm. He showed how a small amount of secret shared randomness (the seed for a pseudorandom generator) could be used to communicate at the Shannon capacity over any polynomial-time channel that introduces a bounded number of errors. Micali *et al.* [21] gave a similar result in a public key model; however, their result relies on efficiently list-decodable codes, which are only known with sub-optimal rate. Both results assume the existence of one-way functions and some kind of setup. On the positive side, in both cases the channel's time bound need not be known explicitly ahead of time; one gets a trade-off between the channel's time and its probability of success.

Our list decoding result removes the setup assumptions of [20], [21] at the price of imposing a specific polynomial bound on the channel's running time and relaxing to list-decoding. However, our result also implies stronger *unique decoding* results in the public-key model [21]. Specifically, our codes can be plugged into the construction of Micali *et al.* to get unique decoding at rates up to the Shannon capacity when the sender has a public key known to the decoder (and possibly to the channel). The idea, roughly, is to sign messages before encoding them; see [21] for details.

Logarithmic-space channels: Galil *et al.* [9] considered a slightly weaker model, logarithmic space, that still captures most physically realizable channels. They modeled the channel as a finite automaton with polynomially-many states. Using Nisan's generator for log-space machines [22], they removed the assumption of one-way functions from Lipton's construction in the shared randomness model [20].

We add nonuniformity to their model to get a common generalization of arbitrarily varying channels. Our code construction for logarithmic-space channels removes the assumption of shared setup in the model of [9], at but achieves only list decoding. This relaxation is necessary for some parameter ranges, since unique decoding in this model is impossible when $p > 1/4$.

III. Statements of Results

Recall the notion of stochastic codes: A stochastic binary code of rate $R \in (0,1)$ and block length N is given by an encoding function $\mathsf{Enc} : \{0,1\}^{RN} \times \{0,1\}^b \to \{0,1\}^N$ which encodes the RN message bits, together with some additional random bits, into an N-bit codeword. Here, N and b are integers, and we assume for simplicity that RN is an integer. Throughout the paper, N will denote the block length of the *final* code we are interested in.

A. Codes for worst-case additive errors

Existential result via list decoding. We give a novel construction of stochastic codes for additive errors by combining *linear* list-decodable codes with a certain kind of authentication code called algebraic manipulation detection (AMD) codes. Such AMD codes can detect *additive corruption* with high probability, and were defined and constructed for cryptographic applications in [1]. The linearity of the list-decodable code is therefore crucial to make the combination with AMD codes work. The linearity ensures that the spurious messages output by the list-decoder are all additive offsets of the true message and depend only on the error vector (and not on m, r). An additional feature of our construction is that even when the fraction of errors exceeds p, the decoder outputs a decoding failure with high probability (rather than decoding incorrectly). This feature is important when using these codes as a component in our explicit construction, mentioned next.

The formal result is stated below. Due to space limitations, details of the proof are omitted and can be found in the full version [13]. The notation $\Omega_{p,\varepsilon}$ expresses an asymptotic lower bound in which p and ε are held constant.

Theorem 1. *For every p, $0 < p < 1/2$ and every $\varepsilon > 0$, there* exists *a family of stochastic codes of rate $R \geqslant 1 - H(p) - \varepsilon$ and a deterministic (exponential time) decoder $\mathsf{Dec} : \{0,1\}^N \to \{0,1\}^{RN} \cup \{\perp\}$ such that for every $m \in \{0,1\}^{RN}$ and every error vector $e \in \{0,1\}^N$ of Hamming weight at most pN, $\Pr_r\big[\mathsf{Dec}\big(\mathsf{Enc}(m,r) + e\big) = m\big] \geqslant 1 - 2^{-\Omega_{\varepsilon,p}(N)}$. Moreover, when more than a fraction p of errors occur, the decoder is able to detect this and report a decoding failure ($\perp$) with probability at least $1 - 2^{-\Omega_{\varepsilon,p}(N)}$.*
Given an explicit *family of linear binary codes of rate R that can be efficiently list-decoded from fraction p of errors with list-size bounded by a polynomial function in N, one can construct an* explicit *stochastic code of rate $R - o(1)$ with the above guarantee along with an* efficient *decoder.*

Explicit, efficient codes achieving capacity. Explicit binary list-decodable codes of optimal rate are not known, so one cannot use the above connection to construct *explicit* stochastic codes of rate $\approx 1 - H(p)$ for pN additive errors. Nevertheless, we give an explicit construction of capacity-achieving stochastic codes against worst-case additive errors. The construction is described at a high-level in Section IV and in further detail in Section V.

Theorem 2. *For every $p \in (0,1/2)$, every $\varepsilon > 0$, and infinitely many N, there is an explicit, efficient stochastic code of block length N and rate $R \geqslant 1 - H(p) - \varepsilon$ which corrects a p fraction of additive errors with probability $1 - o(1)$. Specifically, there are polynomial time algorithms* Enc *and* Dec *such that for* every *message $m \in \{0,1\}^{RN}$ and* every *error vector e of Hamming weight at most pN, we have* $\Pr_r[\mathsf{Dec}(\mathsf{Enc}(m;r)+e) = m] = 1 - \exp(-\Omega_\varepsilon(N/\log^2 N))$.

A slight modification of our construction gives codes for the "average error criterion," in which the code is deterministic but the message is assumed to be uniformly random and unknown to the channel.

B. Codes for online log-space bounded channels

We generalize the model of Galil *et al.* [9] to capture both finite automaton-based models as well as arbitrarily varying channels. To model channels (as opposed to Boolean functions), we augment standard branching programs with the ability to output bits at each step.

Definition 1 (Space bounded channels)**.** *An* online space-S channel *is a read-once branching program of width $\leqslant 2^S$ that outputs one bit at each computation step. Specifically, let $Q = \{0,1\}^S$ be a set of 2^S states. For input length N, the channel is given by a sequence of N transition functions $F_i : Q \times \{0,1\} \to Q \times \{0,1\}$, for $i = 1$ to N, along with a start state $q_0 \in Q$. On input $x = (x_1 x_2 \cdots x_N) \in \{0,1\}^N$, the channel computes $(q_i, y_i) = F_i(q_{i-1}, x_i)$ for $i = 1$ to N. The output of the channel, denoted $A(x)$, is $y = (y_1 y_2 \cdots y_N) \in \{0,1\}^N$.*

A randomized online space-S channel *is a probability distribution over the space of deterministic online space-S channels. For a given input x, such a channel induces a corresponding distribution on outputs. A randomized channel $\mathcal{A}$ is* pN-bounded with probability $1-\beta$ *if, for all inputs $x \in \{0,1\}^N$, with probability at least $1-\beta$, the channel flips fewer than pN bits of x, that is,* $\Pr_{A \in \mathcal{A}}[\text{weight}(x \oplus A(x)) > pN] \leqslant \beta$.

We exhibit a very simple "zero space" channel that rules out achieving any positive rate (i.e., the capacity is zero) when $p > 1/4$. In each position, the channel either leaves the transmitted bit alone, sets it to 0, or sets it to 1. The channel works by "pushing" the transmitted codeword towards a different valid codeword (selected at random). This simple channel adds at most $n/4$ errors *in expectation*. We can get a channel with a hard bound on the number of errors by allowing it logarithmic space. Our impossibility result can be seen as strengthening a result by Dey *et al.* [5] for online channels in the special case where $p > 1/4$.

Theorem 3 (Unique decoding is impossible for $p > \frac{1}{4}$)**.** *For every pair of randomized encoding/decoding algorithms* $\mathsf{Enc}, \mathsf{Dec}$ *that make N uses of the channel and use a message space whose size tends to infinity with N, for every $0 < \nu < \frac{1}{4}$, there is an online space-$\lceil \log(N) \rceil$ channel W_2 that alters at most $N(\frac{1}{4}+\nu)$ bits and causes a uniformly random message to be incorrectly decoded with probability $\Omega(\nu)$.*

For list-decoding, we provide a positive result, namely, a construction of codes with rate approaching $1 - H(p)$ that efficiently recover a short list containing the correct message when the channel uses logarithmic space. The structure of the code is similar to the uniquely decodable code for additive errors; however, additional work is needed to make the codewords appear pseudorandom to the channel, and the analysis is more subtle.

Theorem 4. *For every $p \in (0,1/2)$ and constant $\varepsilon > 0$, there is an efficient Monte Carlo construction of a stochastic code with encoding/decoding algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *such that for every message $m \in \{0,1\}^{(1-H(p)-\varepsilon)N}$ and every randomized online space-S channel W_S on N input bits that is pN-bounded (where $\Omega(\log N) \leqslant S \leqslant o(N/\log N)$), with high probability over the choice of coins r and the errors introduced by W_S, $\mathsf{Dec}(\mathsf{W}_S(\mathsf{Enc}(m;r)))$ outputs a list of at most $\text{poly}(1/\varepsilon)$ messages that includes the real message m.*

The probability of incorrect decoding is at most $N2^{-\Omega(\varepsilon^3 S)} + 2^{-\Omega(\varepsilon^3 N/S)}$, and the running time of $(\mathsf{Enc}, \mathsf{Dec})$ *is is polynomial in N and 2^S (and therefore polynomial in N for log-space channels).*

C. List-decoding for Time-bounded Channels

Finally, we prove a similar result for time-bounded channels, assuming the existence of certain pseudorandom generators (which in turn follow from standard complexity assumptions). The model here is easy to describe: it suffices that the channel be implementable by a circuit of size N^c for some $c \geqslant 1$.

Theorem 5. *Assume either $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{\varepsilon_0 n})$ for some $\varepsilon_0 > 0$ or the existence of one-way functions. For all constants $\varepsilon > 0$, $p \in (0,1/2)$, and $c \geqslant 1$, and for infinitely many integers N, there exists a Monte Carlo* *construction (succeeding with probability $1 - N^{-\Omega(1)}$) of a stochastic code of block length N and rate $R \geqslant 1 - H(p) - \varepsilon$ with $N^{O(c)}$ time encoding/list decoding algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *that have the following property: For all messages $m \in \{0,1\}^{RN}$, and all pN-bounded channels* W *that are implementable by a size $O(N^c)$ circuit,* $\mathsf{Dec}(\mathsf{W}(\mathsf{Enc}(m;r)))$ *outputs a list of at most* $\text{poly}(1/\varepsilon)$ *messages that includes the real message m with probability at least $1 - N^{-\Omega(1)}$.*

The proofs of Theorems 4 and 5 can be found in the full version [13]. In the remainder of the paper, we describe the high level approach behind our code constructions, and present some details of the proof of Theorem 2 on correcting additive errors.

IV. Construction overview

Codes for Additive Errors: Our result is obtained by combining several ingredients from pseudorandomness and coding theory. At a high level the idea (introduced by Lipton [20] in the context of shared randomness) is that if we permute the symbols of the codewords randomly *after* the error pattern is fixed, then the adversarial error pattern looks random to the decoder. Therefore, an explicit code C_{BSC} that can achieve capacity for the binary symmetric channel (such as Forney's concatenated code [8]) can be used to communicate on ADV_p after the codeword's symbols are randomly permuted. This allows one to achieve capacity against adversarial errors when the encoder and decoder share randomness that is unknown to the adversary causing the errors. But, crucially, this requires the decoder to know the random permutation used for encoding.

Our encoder communicates the random permutation (in encoded form) also as part of the overall codeword, without relying on any shared randomness, public key, or other "extra" information. The decoder must be able to figure out the permutation correctly, based solely on a noisy version of the overall codeword (that encodes the permutation plus the actual data). The seed used to pick this random permutation (plus some extra random seeds needed for the construction) is encoded by a low rate code that can correct several errors (say, a Reed-Solomon code) and this information is dispersed into randomly located blocks of the overall codeword (see Figure 1). The locations of the control blocks are picked by a "sampler" — the seed for this sampler is also part of the *control information* along with the seed for the random permutation.

The key challenge is to ensure that the decoder can figure out which blocks encode the control information, and which blocks consist of "data" bits from the codeword of C_{BSC} (the "payload" codeword) that encodes the actual message. The control blocks (which comprise a tiny portion of the overall codeword) are further encoded by a stochastic code (call it the *control code*) that can correct somewhat more than a fraction p, say a fraction $p + \varepsilon$, of errors. These codes can have any constant rate — since they encode a small portion of the message their rate is not so important, so we can use explicit sub-optimal codes for this purpose.

Together with the random placement of the encoded control blocks, the control code ensures that a reasonable ($\Omega(\varepsilon)$) fraction of the control blocks (whose encodings by the control code incur fewer than $p + \varepsilon$ errors) will be correctly decoded. Moreover, blocks with too many errors will be flagged as erasures with high probability. The fraction of correctly recovered control blocks will be large enough that all the control information can be recovered by decoding the Reed-Solomon code used to encode the control information into these blocks. This recovers the permutation used to scramble the symbols of the concatenated codeword. The decoder can then unscramble the symbols in the message blocks and run the standard algorithm for the concatenated code to recover the message.

One pitfall in the above approach is that message blocks could potentially get mistaken for corrupted control blocks and get decoded as erroneous control information that leads the whole algorithm astray. To prevent this, in addition to scrambling the symbols of the message blocks by a (pseudo)random permutation, we also add a pseudorandom offset (which is nearly t-wise independent for some t much larger than the length of the blocks). This will ensure that with high probability each message block will be very far from every codeword and therefore will not be mistaken for a control block.

An important issue we have glossed over is that a uniformly random permutation of the n bits of the payload codeword would take $\Omega(n \log n)$ bits to specify. This would make the control information too big compared to the message length; we need it to be a tiny fraction of the message length. We therefore use almost t-wise independent permutations for $t \approx \varepsilon n / \log n$. Such permutations can be sampled with $\approx \varepsilon n$ random bits. We then make use of the fact that C_{BSC} enables reliable decoding even when the error locations have such limited independence instead of being a uniformly random subset of all possible locations [25].

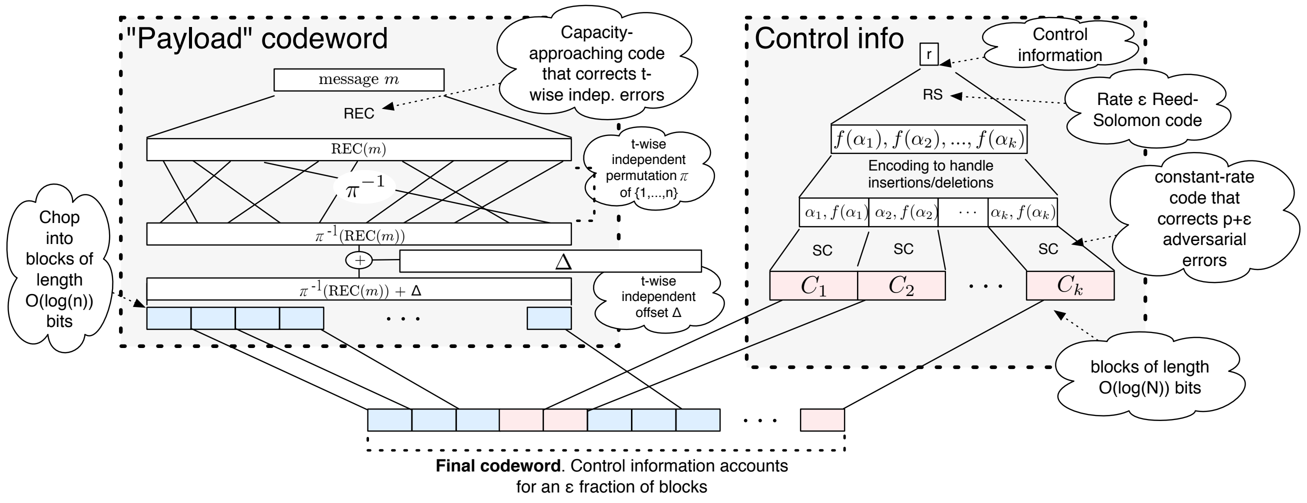
"Payload" codeword
message m
REC
Capacity-approaching code that corrects t-wise indep. errors
REC(m)
π^{-1}
t-wise independent permutation π of {1,...,n}
π^{-1}(REC(m))
Δ
π^{-1}(REC(m)) + Δ
t-wise independent offset Δ
Chop into blocks of length O(log(n)) bits
Control info
r
Control information
RS
Rate ε Reed-Solomon code
$f(\alpha_1), f(\alpha_2), ..., f(\alpha_k)$
Encoding to handle insertions/deletions
$\alpha_1, f(\alpha_1)$
$\alpha_2, f(\alpha_2)$
$\alpha_k, f(\alpha_k)$
SC
constant-rate code that corrects p+ε adversarial errors
C_1
C_2
C_k
blocks of length O(log(N)) bits
Final codeword. Control information accounts for an ε fraction of blocks

Figure 1. Schematic description of encoder from Algorithm 1.

Extending the Construction to log-space and poly-time channels: The construction for additive channels does not work against more powerful channels for (at least) two reasons:

(i) A more powerful channel may inject a large number of correctly formatted control blocks into the transmitted word (recall, each of the blocks is quite small). Even if the real control blocks are uncorrupted, the decoder will have trouble determining which of the correct-looking control blocks is in fact legitimate.

(ii) Since the channel can decide the errors after seeing parts of the codeword, it may be able to learn which blocks of the codeword contain the control information and concentrate errors on those blocks. Similarly, we have to ensure that the channel does not learn about the permutation used to scramble the payload codeword and thus cause a bad error pattern that cannot be decoded by the standard decoder for the concatenated code.

The first obstacle is the easier one to get around, and we do so by using list-decoding: although the channel may inject spurious possibilities for the control information, the total number of such spurious candidates will be bounded. This ensures that after list decoding, provided at least a small fraction of the true control blocks do not incur too many errors, the list of candidates will include the correct control information with high probability.

To overcome the second obstacle, we make sure, using appropriate pseudorandom generators and employing a "hybrid" argument, that the encoding of the message is indistinguishable from a random string by a computationally limited channel (such as one restricted to be online log-space or polynomial time bounded), even when the channel has knowledge of the message and certain parts of the control information. (For concreteness, let us focus on the online log-space case in the following discussion.) This ensures that the distribution of errors caused by the channel on the codeword is indistinguishable by online log-space tests from the distribution caused by the channel on a uniformly random string. Note that the latter distribution is oblivious to the codeword. If these error distributions were in fact *statistically close* (and not just close w.r.t. online log-space tests), successful decoding under oblivious errors would also imply successful decoding under the error distribution caused by the online log-space channel.

The condition that enough control blocks have at most a fraction $p + \varepsilon$ of errors can be checked in online log-space given non-uniform knowledge of the location of the control blocks. We use this together with the above indistinguishability to first prove that enough control blocks are correctly list-decoded, and thus the correct control information is among the candidates obtained by list decoding the control code. Towards showing that the payload codeword is correctly decoded given knowledge of the correct control information, we argue that certain events that imply successful decoding of the concatenated payload codeword, and which we showed to occur with high probability against oblivious errors, also happen with good probability against errors caused by the online log-space channel. The natural approach towards this is to show that assuming this is not the case, one can construct an online log-space distinguisher for the error distributions thereby contradicting their computational indistinguishability. Indeed, this can essentially be shown in the case of polynomial-time bounded channels. This part is harder for the space-bounded case, since Nisan's generator only ensures that the error distribution caused by the channel is indistinguishable from oblivious errors by *online* space-bounded machines. However, the unscrambling of the error vector (according to the permutation that was applied to the payload codeword) cannot be done in an online fashion. So we have to resort an indirect argument based on showing limited independence of certain events related to the payload decoding.

V. Explicit Codes of Optimal Rate for Additive Errors

This section provides a brief sketch of our construction of codes for additive errors. Details and full proofs appear in the full version [13].

A. Ingredients

Our construction uses a number of tools from coding theory and pseudorandomness.

- A constant-rate explicit stochastic code $\mathsf{SC} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^{c_0 b}$, defined on blocks of length $c_0 b = \Theta(\log N)$, that is efficiently decodable with probability $1 - c_1/N$ from a fraction $p + O(\varepsilon)$ of *additive* errors. decodable with probability $1 - c_1/N$. These codes are obtained via Theorem 1.
- A rate $O(\varepsilon)$ Reed-Solomon code RS which encodes a message as the evaluation of a polynomial at points $\alpha_1, ..., \alpha_\ell$ in such a way that an efficient algorithm RS-Decode can efficiently recover the message given at most $\varepsilon\ell/4$ correct symbols and at most $\varepsilon/24$ incorrect ones.
- A randomness-efficient *sampler* $\mathsf{Samp} : \{0,1\}^\sigma \to [N]^\ell$, such that for any subset $B \subseteq [N]$ of size at least μN, the output set of the sampler intersects with B in roughly a μ fraction of its size, that is $|\mathsf{Samp}(s) \cap B| \approx \mu|\mathsf{Samp}(s)|$, with high probability over $s \in \{0,1\}^\sigma$ (*e.g.*, due to Vadhan [26]).
- A generator $\mathsf{KNR} : \{0,1\}^\sigma \to S_n$ for an (almost) t-wise independent family of permutations of the set $\{1, ..., n\}$, that uses a seed of $\sigma = O(t \log n)$ random bits (Kaplan, Naor, and Reingold [16]).
- A generator $\mathsf{POLY}_t : \{0,1\}^\sigma \to \{0,1\}^n$ for a t-wise independent distribution of bit strings of length n, that uses a seed of $\sigma = O(t \log n)$ random bits.
- An explicit efficiently decodable, rate $R = 1 - H(p) - O(\varepsilon)$ code $\mathsf{REC} : \{0,1\}^{Rn} \to \{0,1\}^n$ that can correct a p fraction of t-wise independent errors, that is: for every message $m \in \{0,1\}^{Rn}$, and every error vector $e \in \{0,1\}^n$ of Hamming weight at most pn, we have REC-Decode$(\mathsf{REC}(m) + \pi(e)) = m$ with probability at least $1 - 2^{-\Omega(\varepsilon^2 t)}$ over the choice of a permutation $\pi \in_R \mathrm{range}(\mathsf{KNR})$. (Here $\pi(e)$ denotes the permuted vector: $\pi(e)_i = e_{\pi(i)}$.) A standard family of concatenated codes satisfies this property (Smith [25]).

B. Analysis

The encoding algorithm is given in Algorithm 1 (page 9). The corresponding decoder is given in Algorithm 2 (page 9). Also, a schematic illustration of the encoding is in Figure 1. The reader might find it useful to keep in mind the high level description from Section IV when reading the formal description.

First, note that the rate R of the overall code approaches the Shannon bound: R is almost equal to the rate R' of the code REC used to encode the actual message bits m, since the encoded control information has length $O(\varepsilon N)$. The code REC needs to correct a fraction $p + 25\Lambda\varepsilon$ of t-wise independent errors, so we can pick $R' \geqslant 1 - H(p) - O(\varepsilon)$. Now the rate $R = \frac{R'N'}{N} = R'(1 - 24\Lambda\varepsilon) \geqslant 1 - H(p) - O(\varepsilon)$ (for small enough $\varepsilon > 0$).

We now turn to the analysis of the decoder. Fix a message $m \in \{0,1\}^{R \cdot N}$ and an error vector $e \in \{0,1\}^N$ with Hamming weight at most pN. Suppose that we run Enc on m and coins ω chosen *independently* of the pair m, e, and let $x = \mathsf{Enc}(m; \omega) + e$. The decoder parses x into blocks $x_1, ..., x_{n'+\ell}$ of length $\Lambda \log N$, corresponding to the blocks output by the encoder.

The three lemmas below, proved in the full version, show that the decoder recovers the control information correctly with high probability. We then sketch why the payload message is correctly recovered.

The lemmas illuminate the roles of the main pseudo-random objects in the construction. First, the sampler seed is used to ensure that errors are not concentrated on the control blocks. We say a sampled set T is *good* for error vector e if the fraction of control blocks with relative error rate at most $p + \varepsilon$ is at least $\frac{\varepsilon}{2}$.

Lemma 6 (Good sampler lemma)**.** *For any error vector e of relative weight at most p, with probability at least $1 - \exp(-\Omega(\varepsilon^3 N / \log N)$ over the choice of sampler seed s_T, the set T is* good *for e.*

Given a good sampler seed, the properties of the stochastic code SC guarantee that many control blocks are correctly interpreted. Specifically:

Algorithm 1. ENCODE: On input parameters N, p, ε (with $p + \varepsilon < 1/2$), and message $m \in \{0,1\}^{R \cdot N}$, where $R = 1 - H(p) - O(\varepsilon)$.

```
1: $\Lambda \leftarrow 2c_0$    ▷ Here $c_0$ is the expansion of the stochastic code from Theorem 1 that can correct a fraction $p + \varepsilon$ of errors.
   $n \leftarrow \frac{N}{\Lambda \log N}$    ▷ The final codeword consists of $n$ *blocks* of length $\Lambda \log N$.
   $\ell \leftarrow 24\varepsilon N / \log N$    ▷ The control codeword is $\ell$ blocks long.
   $n' \leftarrow n - \ell$ and $N' \leftarrow n' \cdot (\Lambda \log N)$    ▷ The payload codeword is $n'$ blocks long (i.e. $N'$ bits).
2: **Select seeds** $s_\pi, s_\Delta, s_T$ uniformly in $\{0,1\}^{\varepsilon^2 N}$.
   Set the *control information* $\omega$ to be the concatenation $(s_\pi, s_\Delta, s_T)$.
3: **Encode** $\omega$ **with a Reed-Solomon code** RS to get symbols $(a_1, ..., a_\ell)$.
       ▷ RS is a rate $\frac{\varepsilon}{8}$ Reed-Solomon code which evaluates polynomials at points $(\alpha_1, \ldots, \alpha_\ell)$ in a field $\mathbb{F}$ of size $\approx N$.
4: **Encode each symbol together with its evaluation point**: For $i = 1, ..., \ell$, do
   • $A_i \leftarrow (\alpha_i, a_i)$
   • $C_i \leftarrow \mathsf{SC}(A_i, r_i)$, where $r_i$ is random of length $2 \log N$ bits.    ▷ SC corrects additive errors with high probability.
5: **Encode** $m$ **using a code that corrects *random* errors**:
   • $P \leftarrow \mathsf{REC}(m)$,    ▷ REC : $\{0,1\}^{R'N'} \to \{0,1\}^{N'}$ is a code that corrects a $p + 25\Lambda\varepsilon$ fraction of *t-wise independent* errors. Here $R' = \frac{RN}{N'}$.
6: **Expand the seeds** $s_T, s_\Delta, s_\pi$ to get a set $T = \mathsf{Samp}(s_T)$, offset $\Delta = \mathsf{POLY}(s_\Delta)$, and permutation $\pi = \mathsf{KNR}(s_\pi)$.
7: **Scramble the payload codeword:**
   • $\pi^{-1}(P) \leftarrow$ (bits of $P$ permuted according to $\pi^{-1}$)
   • $Q \leftarrow \pi^{-1}(P) \oplus \Delta$
   • Cut $Q$ into $n'$ blocks $B_1, ... B_{n'}$ of length $\Lambda \log N$ bits.
8: **Interleave** control blocks $C_1, ..., C_\ell$ with payload blocks $B_1, ..., B_{n'}$, using control blocks in positions from $T$ and payload blocks in remaining positions.
```

Algorithm 2. DECODE: On input x of length N:

```
1: Cut $x$ into $n' + \ell$ blocks $x_1, ..., x_{n'+\ell}$ of length $\Lambda \log(n)$ each.
2: **Attempt to decode control blocks**: For $i = 1, ..., n' + \ell$, do
   • $\tilde{F}_i \leftarrow$ SC-DECODE$(x_i)$.
       ▷ With high prob, non-control blocks are rejected (Lemma 8), and control blocks are either correctly decoded or discarded (Lemma 7).
   • If $\tilde{F}_i \neq \perp$, then parse $\tilde{F}_i$ as $(\tilde{\alpha}_i, \tilde{a}_i)$, where $\tilde{\alpha}_i, \tilde{a}_i \in \mathbb{F}$.
3: $(\tilde{s}_T, \tilde{s}_\Delta, \tilde{s}_\pi) \leftarrow$ RS-DECODE$\Big($pairs $(\tilde{\alpha}_i, \tilde{a}_i)$ output above$\Big)$.    ▷ Control information is recovered w.h.p.
4: Expand the seeds $\tilde{s}_T, \tilde{s}_\Delta, \tilde{s}_\pi$ to get set $\tilde{T}$, offset $\tilde{\Delta}$, and permutation $\tilde{\pi}$.
5: $\tilde{Q} \leftarrow$ concatenation of blocks $x_i$ not in $\tilde{T}$
6: $\tilde{P} \leftarrow \pi(\tilde{Q} \oplus \tilde{\Delta})$    ▷ If control info is correct, then errors in $\tilde{P}$ are almost $t$-wise independent.
7: $\tilde{m} \leftarrow$ REC-DECODE$(\tilde{P})$
```

Lemma 7 (Control blocks lemma)**.** *For all e, T such that T is good for e, with probability at least $1 - \exp(-\Omega(\varepsilon^3 N / \log N))$ over the random coins $(r_1, r_2, \ldots, r_\ell)$ used by the ℓ SC encodings, we have: 1) The number of control blocks correctly decoded by* SC-DECODE *is at least $\frac{\varepsilon \ell}{4}$. 2) The number of* erroneously *decoded control blocks is less than $\frac{\varepsilon \ell}{24}$. (By erroneously decoded, we mean that* SC-DECODE *outputs neither $\perp$ nor the correct message.)*

The offset Δ is then used to ensure that payload blocks are not mistaken for control blocks:

Lemma 8 (Payload blocks lemma)**.** *For all m, e, s_T, s_π, with probability at least $1 - 2^{-\Omega(\varepsilon^2 N / \log^2 N))}$ over the offset seed s_Δ, the number of* payload *blocks incorrectly accepted as control blocks by* SC-DECODE *is less than* $\frac{\varepsilon\ell}{24}$.

The two previous lemmas imply that the Reed-Solomon decoder will, with high probability, be able to recover the control information.

It remains to analyze the final decoding process. First, suppose that the correct control information $\omega = (s_\pi, s_\Delta, s_T)$ is handed directly to the decoder— *i.e.*, assume we are in the "shared randomness" setting. Fix m, e, and s_T, and let e_Q be the restriction of the error e to the payload codeword. The relative weight of e_Q is at most $\frac{pN}{N'} \leqslant p(1 + 25\Lambda\varepsilon)$ for sufficiently small ε. Consider the string $\tilde{P}$ that is input the the REC decoder. We can write $\tilde{P} = \tilde{\pi}(\tilde{Q} \oplus \tilde{\Delta}) = \pi(Q \oplus e_Q \oplus \Delta) = P \oplus \pi(e_Q)$. Since s_π is selected independently from T, the permutation π is independent of the payload error e_Q. and so the input to REC is corrupted by at most $p(1+25\Lambda\varepsilon)N'$ errors which are t-wise independent, for appropriate t. By the properties of REC, with probability at least $1 - e^{-\Omega(\varepsilon^4 N/\log N)}$, the message m is correctly recovered by DECODE.

The real error probability is slightly higher since we must condition on correct recovery of the control information; see the full version for details.

REFERENCES

[1] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT*, pages 471–488, 2008.

[2] I. Csiszár and P. Narayan. Arbitrarily varying channels with constrained inputs and states. *IEEE Transactions on Information Theory*, 34(1):27–34, 1988.

[3] I. Csiszár and P. Narayan. The capacity of the arbitrarily varying channel revisited: Positivity, constraints. *IEEE Transactions on Information Theory*, 34(2):181–193, 1988.

[4] I. Csiszár and P. Narayan. Capacity and decoding rules for classes of arbitrarily varying channels. *IEEE Transactions on Information Theory*, 35(4):752–769, 1989.

[5] B. K. Dey, S. Jaggi, and M. Langberg. Codes against online adversaries. *CoRR*, abs/0811.2850, 2008.

[6] P. Elias. List decoding for noisy channels. Technical Report 335, Research Lab. of Electronics, MIT, 1957.

[7] P. Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37:5–12, 1991.

[8] G. D. Forney. *Concatenated Codes*. MIT Press, Cambridge, MA, 1966.

[9] Z. Galil, R. J. Lipton, X. Yu, and M. Yung. Computational error-correcting codes achieve shannon's bound explicitly. *Manuscript*, 1995.

[10] V. Guruswami. List decoding with side information. In *Conference on Computational Complexity (CCC)*, pages 300–309, July 2003.

[11] V. Guruswami. Algorithmic Results in List Decoding. *Foundations and Trends in Theoretical Computer Science (FnT-TCS)*, 2(2), January 2007.

[12] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1035, 2002.

[13] V. Guruswami and A. Smith. Codes for computationally simple channels. arXiv.org eprint 1004.4017, 2010.

[14] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[15] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on the Theory of Computing*, pages 220–229, 1997.

[16] E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1): 113–133 (2009). Preliminary version in *RANDOM 2005*.

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

[18] M. Langberg. Oblivious communication channels and their capacity. *IEEE Transactions on Information Theory*, 54(1):424–429, 2008.

[19] A. Lapidoth and P. Narayan. Reliable communication under channel uncertainty. *IEEE Transactions on Information Theory*, 44(6):2148–2177, 1998.

[20] R. J. Lipton. A new approach to information theory. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 699–708, 1994.

[21] S. Micali, C. Peikert, M. Sudan, and D. A. Wilson. Optimal error correction against computationally bounded noise. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 1–16, 2005.

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

[23] N. Nisan and A. Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[24] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[25] A. Smith. Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In *Symposium on Discrete Algorithms*, pages 395–404, 2007.

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

[27] J. M. Wozencraft. List Decoding. *Quarterly Progress Report, Research Lab. of Electronics, MIT*, 48:90–95, 1958.

[28] A. C.-C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[29] V. V. Zyablov and M. S. Pinsker. List cascade decoding. *Problems of Information Transmission*, 17(4):29–34, 1981 (in Russian); pp. 236-240 (in English), 1982.