Towards Optimal Deterministic Coding for Interactive Communication

Ran Gelles* Bernhard Haeupler† Gillat Kol‡ Noga Ron-Zewi§ Avi Wigderson¶

Abstract

We study *efficient, deterministic* interactive coding schemes that simulate any interactive protocol both under random and adversarial errors, and can achieve a constant communication rate independent of the protocol length.

For channels that flip bits independently with probability $\epsilon < 1/2$, our coding scheme achieves a communication rate of $1 - O(\sqrt{H(\epsilon)})$ and a failure probability of $\exp(-n/\log n)$ in length n protocols. Prior to our work, all nontrivial deterministic schemes (either efficient or not) had a rate bounded away from 1. Furthermore, the best failure probability achievable by an efficient deterministic coding scheme with constant rate was only quasi-polynomial, i.e., of the form $\exp(-\log^{O(1)} n)$ (Braverman, ITCS 2012).

For channels in which an adversary controls the noise pattern our coding scheme can tolerate $\Omega(1/\log n)$ fraction of errors with rate approaching 1. Once more, all previously known nontrivial deterministic schemes (either efficient or not) in the adversarial setting had a rate bounded away from 1, and no nontrivial efficient deterministic coding schemes were known with any constant rate.

Essential to both results is an explicit, efficiently encodable and decodable *systematic tree code* of length n that has relative distance $\Omega(1/\log n)$ and rate approaching 1, defined over an $O(\log n)$-bit alphabet. No nontrivial tree code (either efficient or not) was known to approach rate 1, and no nontrivial distance bound was known for any efficient constant rate tree code. The fact that our tree code is *systematic*, turns out to play an important role in obtaining rate $1 - O(\sqrt{H(\epsilon)})$ in the random error model, and approaching rate 1 in the adversarial error model.

A central contribution in deriving our coding schemes for random and adversarial errors, is a novel code-concatenation scheme, a notion standard in coding theory which we adapt for the interactive setting. We use the above tree code as the "outer code" in this concatenation. The necessary deterministic "inner code" is achieved by a nontrivial derandomization of the randomized interactive coding scheme of (Haeupler, STOC 2014). This deterministic coding scheme (with *exponential* running time, but applied here to $O(\log n)$ bit blocks) can handle an ϵ fraction of adversarial errors with a communication rate of $1 - O(\sqrt{H(\epsilon)})$.

1 Introduction

1.1 Background

Coding for interactive communication, the subject of this paper, connects two large bodies of work, coding theory and communication complexity. Both study communication cost, but with very different settings and goals in mind. Coding Theory, born with Shannon's and Hamming's breakthrough papers [23, 16], is a vast discipline which deals largely with *one-way* communication between two remote parties (Alice and Bob), each holding an input (resp. x, y, possibly from some joint distribution). Major focus is on a *single* communication task: Alice wants to convey x to Bob, and the challenge is doing so reliably when the communication channel between them is *unreliable*, e.g. some of the communicated bits are flipped randomly or adversarially. Alice's messages are encoded by longer codewords to overcome this "noise", and one attempts to minimize the communication cost needed to achieve high reliability in each noise model. Communication Complexity, an important research area introduced by Yao [25] 30 years later, also strives to minimize communication cost, but has an opposite focus: it assumes a *perfect* communication channel between Alice and Bob, who now want to perform an *arbitrary* communication task (e.g. computing an arbitrary function $f(x, y)$) using a *two-way* interactive communication protocol.

The seminal work of Schulman [19, 20, 21] merged these two important subjects, and studied coding schemes for arbitrary two-way interactive communication protocols. Given the interaction and adaptive nature of two-way protocols, this significantly extends the

*Department of Computer Science, Princeton University, rgelles@cs.princeton.edu. Supported in part by NSF grant CCF-1149888.

†Carnegie Mellon University, haeupler@cs.cmu.edu

‡Institute for Advanced Study, Princeton, NJ. Research at the IAS supported by The Fund For Math and the Weizmann Institute of Science National Postdoctoral Award Program for Advancing Women in Science.

§School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA and DIMACS, Rutgers University, Piscataway, NJ, USA. This research was partially supported by NSF grants CCF-1412958 and CCF-1445755 and the Rothschild fellowship. nogazewi@ias.edu

¶Institute for Advanced Study, Princeton, USA, avi@ias.edu. This research was partially supported by NSF grant CCF-1412958.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

challenge of coding theory. For example, while trade-offs between coding parameters, like the fraction of correctable errors and redundancy for one-way communication have been well understood at least in principle already in Shannon's paper, and efficient codes matching them were found for many channels, these questions are still far from understood in the two-way case.

In the above papers Schulman set up the basic models, proved the *existence* of nontrivial coding schemes for any interactive protocol, in both the random and the adversarial noise models, and gave an efficient randomized scheme for random noise. Progress on finding trade-offs between parameters, and approaching them using *efficient* coding schemes has been slow for a while, but the past few years have seen an impressive flood of new techniques and results on the many challenging questions raised by this general setting, see e.g. [6, 10, 3, 17, 1, 2, 4, 12, 11, 14, 9]. To (informally) cite but one central recent result, Kol and Raz [17] (see also [14]) proved that for the binary symmetric channel BSC_ϵ (in which every communicated bit is flipped independently with probability ϵ), the communication rate is $1 - \tilde{\Theta}(\sqrt{H(\epsilon)})$, where H is the binary entropy function (this should be contrasted with the one-way setting in which the communication rate of BSC_ϵ is known to be $1 - H(\epsilon)$).

Let us describe the basic protocol structure and coding problem a bit more precisely, as in this young field models vary, and some choices actually affect the basic parameters. Many different variants are discussed in many different works but for our focus the assumptions we make are quite standard. Assume that Alice and Bob communicate to perform a distributed task, e.g. compute some function $f(x, y)$ on their respective private inputs x and y. We fix a communication protocol π for this task, in which we assume that the parties alternate: Alice sends one bit in odd steps, and Bob sends one bit in even steps. We further assume that they communicate the same number of bits on every input (the length $n = |\pi|$ of π will be our main complexity parameter). Finally, we assume that each party outputs $\pi(x, y)$, the entire transcript of their conversation (this "universal" output captures all possible tasks, e.g. computing a function). If there is no noise on the channel, this is the standard communication complexity setting.

Now assume that the communication channel is noisy. We consider two main noise models, probabilistic and adversarial. In the probabilistic BSC_ϵ model each communicated bit is flipped independently with a constant probability ϵ. In the adversarial model an adversary can flip an ϵ fraction of the communication bits. To cope with the errors, Alice and Bob run a *coding scheme* Π that should *simulate* the noiseless protocol π over the noisy channel. That is, for any inputs x, y the parties hold and for any noiseless protocol π, after running the coding scheme Π over the noisy channel, each party should output $\pi(x, y)$ (if the coding scheme and/or the channel are probabilistic, this should happen with high probability over the randomness). We assume that the parties alternate also in the execution of Π, and as we assumed π has a fixed length (communication cost) $|\pi|$ for every input, we can assume the same for Π, denoting its length by $|\Pi|$.

One basic parameter of a coding scheme Π is the *rate*, defined as $|\pi|/|\Pi|$, which captures the redundancy of Π relative to the noiseless protocol π (this definition parallels the standard one of rate as the ratio between message length to codeword length in one-way communication). Ideally, the rate should be at least some constant independent of the protocol length $|\pi|$ (and even better, approach the capacity of these channels). Two other important goals are computational efficiency and determinism. Ideally the computational complexity of the parties using the scheme Π (given π and their inputs) should be at most polynomial in $|\pi|$ (or better, near-linear), and the coding scheme should be deterministic.[1] Furthermore, in the probabilistic noise model, we desire the coding scheme to fail with small probability (ideally, exponentially small probability in $|\pi|$) over the randomness of the channel, and in the adversarial model, we would like the fraction of correctable errors ϵ to be as large as possible (ideally, a large constant).[2] Somewhat surprisingly we find a connection between the last two, seemingly unrelated, goals.

1.2 Main results

In this work we focus on the above three goals: constant rate, efficiency and determinism. Achieving all simultaneously with a non-trivial coding scheme is not a simple task. For instance, the schemes of [20, 6, 4] are deterministic, yet inefficient; [19, 10, 17, 2, 12, 11, 14] give efficient coding schemes, yet randomized; [3] provides an efficient and deterministic coding scheme over BSC_ϵ with constant rate, but with a rate bounded away from 1 and with quasi-polynomial failure probability.

Our first main result provides the first interactive coding scheme for BSC_ϵ channels which is both efficient,

[1] This presents a subtle issue in the presence of random noise. To prevent a deterministic coding scheme from using the random noise as a source of randomness, one actually uses the so-called *arbitrarily varying channel* (AVC) that extends BSC_ϵ, with which an adversary may determine the *probability* $\epsilon_i \in [0, \epsilon]$ in which the i-th bit is flipped, see e.g., [7]. In particular, Π must be correct also if there is no noise at all.

[2] The parameter ϵ cannot exceed 1/4, see [6, 4].

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

deterministic and has a rate that approaches 1 as $\epsilon \to 0$. Furthermore, it has nearly exponential failure probability and nearly optimal rate.

THEOREM 1.1. *For every $\epsilon > 0$ there exists an efficient deterministic interactive coding scheme Π that simulates any noiseless protocol π of length n over BSC_ϵ with rate $1 - O\big(\sqrt{H(\epsilon)}\big)$ and failure probability $\exp(-\Omega(\epsilon^4 n/\log n))$.*

Our second main result gives the first efficient and deterministic interactive coding scheme with constant rate (that can even be taken arbitrarily close to 1) over an adversarial channel, albeit with a lower noise rate of $O(1/\log n)$. Improving the fraction of errors to a constant (which can be done by either randomized or inefficient schemes) remains a challenging problem for which this may be a stepping stone.

THEOREM 1.2. *For every $\epsilon > 0$ there exists an efficient deterministic interactive coding scheme Π that simulates any noiseless protocol π of length n with rate $1-\epsilon$ in the presence of up to $\Omega\big(\frac{\epsilon^4}{\log(1/\epsilon)} \cdot \frac{1}{\log n}\big)$ fraction of adversarial errors.*

1.3 Overview of techniques

Our coding schemes exploit the idea of *code concatenation*, which is a very common technique in the one-way coding theory literature [8]. Concatenation usually consists of two separate coding layers: an *inner code* which has optimal parameters yet may be inefficient, and an *outer code* which must be efficient yet may have suboptimal parameters. In the standard concatenation the message is first encoded using the outer code. The resulting codeword is then split into small blocks, say of size $O(\log n)$, and each block is separately encoded by the inner code. By choosing the right parameters such concatenation can result in a code which is both efficient and has optimal parameters.

A similar high level idea was recently used in the interactive coding setting in the work [2] that takes a similar course of splitting the simulated protocol into small chunks, and simulating each chunk individually using an inefficient (deterministic) scheme. Then, a layer of synchronization is added, whose purpose is to ensure that the parties agree on the partial transcript simulated so far, and progress to the next chunk or rewind to a previous chunk accordingly. The synchronization is done by exchanging *random* hash values of the currently simulated transcript. Extremely important in this scheme is the fact that the error in these different synchronization steps is independent.

Towards replacing the random hash values with a deterministic substitute, we take a carful examination of the properties the hash functions should satisfy. A natural idea is to replace the hash values by the "redundant" part of a *systematic* error-correcting code[3], in which the partial transcript simulated so far is the systematic part. A key complication in the interactive setting is that if at some point there is an error in the simulated transcript, then for *every* possible future continuation of the transcript, the hash values should indicate this error. Overcoming this obstacle is precisely the motivation behind Schulman's definition of a *tree code* [20], which ensures a large distance between the encodings of any two different continuations of any fixed prefix (see Section 2.2 for a formal definition). Combining these ideas, we replace the randomized synchronization layer in [2] with the redundant part of an efficiently encodable and decodable *systematic* tree code.

Viewing the systematic tree code as the "outer code" in our construction, and the inefficient scheme used to simulate the small chunks as the "inner code", our coding scheme can be alternatively thought of as an instantiation of the standard one-way concatenation method in the interactive coding setting. We stress however that in the interactive coding setting (and unlike the one-way setting), having the outer scheme be systematic is essential for obtaining high rate. Indeed, Asserting that the outer coding scheme would be systematic allows us to run the inner coding scheme as is and view the partial transcript simulated so far as the systematic part of the outer scheme encoding, and then communicate the *only* the redundant part of the outer scheme encoding.

More concretely, as the outer code in our construction we use a novel construction of an efficiently encodable and decodable systematic tree code of depth $\Omega(n/\log n)$, relative distance $\Omega(1/\log n)$ and rate ≈ 1, defined over an $O(\log n)$-bit alphabet. As the inner code we use an exponential time deterministic interactive coding scheme that can correct ϵ fraction of adversarial errors with rate $1-O(\sqrt{H(\epsilon)})$, applied to chunks of length $O(\log n)$ (so that each simulated chunk is a single input symbol of the tree code). We obtain this latter scheme via a derandomization of a randomized scheme due to [14] with similar guarantees. The distance of the tree code guarantees that the entire simulation succeeds as long as at most $O(1/\log n)$ fraction of the inner blocks failed in their simulation. This implies that the scheme succeeds as long as the number of adversarial bit flips is bounded by $O((n/\log n) \times (1/\log n) \times \log n) = O(n/\log n)$; similarly, over BSC_ϵ the simulation

[3] A systematic error-correcting code is one in which the message is a part of the codeword; The message is the systematic part and the rest of the codeword is the redundant part.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

fails with probability at most $\left(2^{-\Omega(\log n)}\right)^{\Omega\left(\frac{n}{\log n}\cdot\frac{1}{\log n}\right)} = 2^{-\Omega(n/\log n)}$.

A by-product of our approach is an interesting connection between the adversarial and random noise settings. Our concatenation lemma (Lemma 3.1) shows that an efficient deterministic (and systematic) tree code that resists δ fraction of adversarial noise (along with the inner code described above), leads to an efficient deterministic coding scheme for BSC_ϵ that succeeds with probability $\approx 1 - 2^{-\Omega(n\delta)}$ over the randomness of the channel. In particular, the failure probability of $2^{-\Omega(n/\log n)}$ over a BSC_ϵ exhibited in Theorem 1.1 follows from the fact that the tree code we use as an outer code has a relative distance of $\delta = O(1/\log n)$. Obtaining an efficient deterministic scheme whose failure probability over BSC_ϵ is *exponentially small* remains an interesting open question. Our concatenation lemma suggests that this question is closely related to the 20-year open question of finding deterministic and efficient tree codes with constant rate and relative distance [20], and obtaining deterministic and efficient coding schemes with constant rate that resist a constant fraction of noise in the adversarial setting.

In what follows we describe the outer and inner codes we use in more detail and the techniques used to obtain them.

Outer code. The high-level idea of the tree code construction is as follows. For every integer t such that $\Omega(\log\log n) \le t \le \log n$ we partition the message to blocks of size 2^t. Each such block is separately encoded via a standard (one-way) systematic error-correcting code with relative distance $\Omega(1/\log n)$ and rate $1 - O(1/\log n)$. This yields a redundant part $R^{(t)}$ of 2^t bits which are layered across the next block, i.e., across the encodings of the next 2^t levels, so that every level gets 1 bit. This layering amortizes the redundancy across the tree, which helps keeping the rate approaching 1 while still providing the required relative distance guarantee of $\Omega(1/\log n)$, yet only over the next 2^t levels. See Figure 1 for an illustration of the construction.

A somewhat related tree code construction is outlined in an unpublished memo by Schulman [22]: that construction uses the same idea of encoding prefixes of increasing lengths 2^t, using an asymptotically good error-correcting code with constant rate and relative distance, then layering the output codeword across the next 2^t levels (a similar high-level idea was used also in [3]). However, since a non-systematic code with rate bounded away from 1 is used this results with rate $O(1/\log n)$; on the other hand, the obtained relative distance is constant $\Omega(1)$. One can view these two constructions as tradeoffing complementary goals: while [22] optimizes the distance at the expense of a low rate, we optimize the rate at the expense of a low distance.

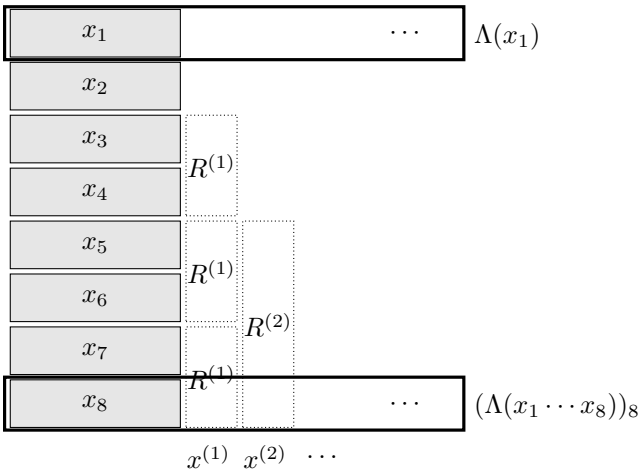

Figure 1: An illustration of the tree code's encoding; see Section 5 for details.

Inner Code. The starting point for the inner code construction is an efficient randomized coding scheme by Haeupler [14]. That scheme achieves rate of $1 - O(\sqrt{\epsilon})$ over a BSC_ϵ, however it is randomized. We obtain our inner code by derandomizing this scheme, and the main technical issue here is to derandomize the scheme without damaging its rate. On the other hand, we do not need the derandomization to be efficient, since we will run the inner code only on protocols of length $O(\log n)$.

Our derandomization approach generalizes the derandomization technique of [2]. More specifically, in [2] the authors show an upper bound of $2^{O(n)}$ on the number of different sequences of partial transcripts that may occur during a run of their coding scheme and then show that for each sequence of transcripts at least $1-2^{-\Omega(\ell\cdot n)}$ of the random strings are good for this sequence, where ℓ is the output length of the hash functions. Choosing ℓ to be a sufficiently large constant, a union bound shows the existence of a single random string that works for all sequences.

In our case, since we want to maintain a high rate we cannot afford the output length of the hash functions to be a too large constant, and so the upper bound of $2^{O(n)}$ is too large for our purposes. To deal with this we show how to relate simulated transcripts to the noise patterns that may have caused them. A fine counting argument reduces the effective number of sequences of transcripts to the number of *typical* noise patterns, i.e., the $2^{O(H(\epsilon)n)}$ patterns with at most ϵ fraction of bit flips. Then a union bound argument on all the possible noise

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

patterns proves the existence of a single random string that works for any typical noise pattern. For this union bound to work we need to use slightly longer hashes than in the original scheme of [14] which results in a slightly reduced rate of $1 - O(\sqrt{H(\epsilon)})$.

1.4 Organization of the paper

We begin (Section 2) by recalling several building blocks and setting up notations we will use throughout. Our main concatenation lemma is provided in Section 3, along with a formal statement of the inner and outer codes we use. These prove our main Theorems 1.1 and 1.2. The detailed proof of the concatenation lemma appears in Section 4. In Sections 5 and 6 we present our inner and outer codes constructions, respectively. Omitted proofs can be found in the full version of this paper.

2 Preliminaries

All logarithms in this paper are taken to base 2. We denote by $H : [0,1] \to [0,1]$ the binary entropy function given by $H(p) = p\log(1/p)+(1-p)\log(1/(1-p))$ for $p \notin \{0,1\}$ and $H(0) = H(1) = 0$. Let $\mathbb{F}_2$ denote the finite field of two elements and let $\mathbb{N}$ denote the set of positive integers. For an integer $n \in \mathbb{N}$ let $[n] := \{1,\ldots,n\}$ and for a pair of integers $m,n \in \mathbb{N}$ such that $m \leq n$ let $[m,n] := \{m,m+1,\ldots,n\}$. For a vector $x \in \Sigma^n$ and integers $1 \leq i \leq j \leq n$ we denote by $x[i,j]$ the projection of x on the coordinates in the interval $[i,j]$, and we let $|x| = n$ denote the length of x. Finally, the *relative distance* between a pair of strings $x,y \in \Sigma^n$ is the fraction of coordinates on which x and y differ, and is denoted by $\mathrm{dist}(x,y) := |\{i \in [n] : x_i \neq y_i\}|/n$.

2.1 Error-correcting codes

A code is a mapping $C : \Sigma_{\mathrm{in}}^k \to \Sigma_{\mathrm{out}}^n$. We call k the *message length* of the code and n the *block length* of the code. The elements in the image of C are called *codewords*. The *rate* of C is the ratio $\frac{k\log|\Sigma_{\mathrm{in}}|}{n\log|\Sigma_{\mathrm{out}}|}$. We say that C has *relative distance* at least δ if for every pair of distinct vectors $x,y \in \Sigma_{\mathrm{in}}^k$ it holds that $\mathrm{dist}(C(x),C(y)) \geq \delta$.

Let $\mathbb{F}$ be a finite field. We say that C is $\mathbb{F}$-*linear* if Σ_{in}, Σ_{out} are vector spaces over $\mathbb{F}$ and the map C is linear over $\mathbb{F}$. If $\Sigma_{\mathrm{in}} = \Sigma_{\mathrm{out}} = \mathbb{F}$ and C is $\mathbb{F}$-linear then we simply say that C is *linear*. Finally, if $k = n$ then we say that a code $C : \Sigma_{\mathrm{in}}^n \to \Sigma_{\mathrm{out}}^n$ is *systematic* if $\Sigma_{\mathrm{in}} = \Gamma^s$ and $\Sigma_{\mathrm{out}} = \Gamma^{s+r}$ for some alphabet Γ and integers $s,r \in \mathbb{N}$, and there exists a string $R(x) \in (\Gamma^r)^n$ such that $(C(x))_i = (x_i,(R(x))_i)$ for every $x \in \Sigma_{\mathrm{in}}^n$ and $i \in [n]$ (that is, the projection of $(C(x))_i$ on the first s coordinates equals x_i). We call x and $R(x)$ the *systematic part* and the *redundant part* of $C(x)$, respectively.

Specific families of codes. We now mention some known constructions of error-correcting codes that we shall use as building blocks in our tree code construction, and state their relevant properties. We start with the following fact that states the existence of Reed-Solomon codes which achieve the best possible trade-off between rate and distance over large alphabets.

Lemma 2.1. (Reed-Solomon codes [18]) *For every $k,n \in \mathbb{N}$ such that $k \leq n$, and for every finite field $\mathbb{F}$ such that $|\mathbb{F}| \geq n$ there exists a linear code* $\mathrm{RS} : \mathbb{F}^k \to \mathbb{F}^n$ *with rate k/n and relative distance at least $1 - \frac{k}{n}$. Furthermore,* RS *can be encoded and decoded from up to $(1-\frac{k}{n})/2$ fraction of errors in time* $\mathrm{poly}(n,\log\mathbb{F})$.

The next lemma states the existence of asymptotically good binary codes. Such codes can be obtained for example by concatenating the Reed-Solomon codes from above with binary linear Gilbert-Varshamov codes [13, 24].

Lemma 2.2. (Asymptotically good binary codes) *For every $0 < \rho < 1$ there exist $\delta > 0$ and integer $k_0 \in \mathbb{N}$ such that the following holds for any integer $k \geq k_0$. There exists a binary linear code $B : \{0,1\}^k \to \{0,1\}^n$ with rate at least ρ and relative distance at least δ. Furthermore, B can be encoded and decoded from up to $\delta/2$ fraction of errors in time* $\mathrm{poly}(n)$.

2.2 Tree codes

A tree code [21] is an error-correcting code $\Lambda : \Sigma_{\mathrm{in}}^n \to \Sigma_{\mathrm{out}}^n$ which is a *prefix-code*: for any $i \in [n]$ and $x \in \Sigma_{\mathrm{in}}^n$ the first i symbols of $\Lambda(x)$ depend only on $x_1,\ldots,x_i$. For simplicity we shall sometimes abuse notation and denote by Λ also the map $\Lambda : \Sigma_{\mathrm{in}}^j \to \Sigma_{\mathrm{out}}^j$ which satisfies that $\big(\Lambda(x_1,\ldots,x_j)\big)_i = \big(\Lambda(x_1,\ldots,x_n)\big)_i$ for every $i \in [j]$ and $x \in \Sigma_{\mathrm{in}}^n$. Observe that this latter map is well defined as $(\Lambda(x_1,\ldots,x_n))_i$ depends only on $x_1,\ldots,x_i$.

We say that Λ has *relative tree distance* at least δ if for every pair of distinct vectors $x,y \in \Sigma_{\mathrm{in}}^n$ such that $i \in [n]$ is the first coordinate on which x and y differ (i.e., $(x_1,\ldots,x_{i-1}) = (y_1,\ldots,y_{i-1})$ but $x_i \neq y_i$), and for every j such that $i \leq j \leq n$ it holds that

$$\mathrm{dist}\Big(\Lambda(x)[i,j],\Lambda(y)[i,j]\Big) \geq \delta.$$

Alternatively, the relative tree distance of a tree code can be defined via the notion of *suffix distance* [9] (see also [4, 5]). The *suffix distance* between a pair of strings

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

$x, y \in \Sigma^n$ is

$$\mathrm{dist}_{\mathrm{sfx}}(x,y) := \max_{i\in[n]}\left\{\mathrm{dist}\Big(x[i,n], y[i,n]\Big)\right\}.$$

It can be shown that a tree code has relative tree distance at least δ if and only if for every pair of distinct vectors $x, y \in \Sigma_{\mathrm{in}}^n$ it holds that $\mathrm{dist}_{\mathrm{sfx}}(\Lambda(x), \Lambda(y)) \geq \delta$

Finally, we say that Λ can be *decoded from α fraction of errors* if there exists an algorithm that is given as input a vector $w \in \Sigma_{\mathrm{out}}^j$ for some $j \in [n]$ and outputs a vector $y \in \Sigma_{\mathrm{in}}^j$ that satisfies the following guarantee: If there exists $x \in \Sigma_{\mathrm{in}}^j$ such that $\mathrm{dist}_{\mathrm{sfx}}\big(\Lambda(x), w\big) \leq \alpha$, then $y = x$.

3 Efficient Deterministic Coding Schemes

In this section we prove our main Theorems 1.1 and 1.2. These theorems are an immediate implication of our concatenation lemma below. The concatenation lemma proves that given an efficient deterministic systematic tree code (used as an outer code) and a possibly inefficient deterministic coding scheme (used as an inner code), one can construct an efficient deterministic coding scheme, and states the parameters of the resulting coding scheme as a function of the parameters of the outer and inner codes. We now give the concatenation lemma (whose proof appears in Section 4). Then, in Lemmas 3.2 and 3.3 below, we state the existence of outer and inner codes with good parameters, whose concatenation proves our main Theorems 1.1 and 1.2.

LEMMA 3.1. (CONCATENATION) *Suppose that the following hold:*

1. *(Outer code) There exists a systematic tree code $\Lambda : \Sigma_{\mathrm{in}}^{n_\Lambda} \to \Sigma_{\mathrm{out}}^{n_\Lambda}$ with $\Sigma_{\mathrm{in}} = \{0,1\}^s$, $\Sigma_{\mathrm{out}} = \{0,1\}^{s+r}$ and rate ρ_Λ that can be encoded and decoded from up to δ_Λ fraction of errors in time $\mathcal{T}_\Lambda$.*
2. *(Inner code) There exists a deterministic interactive coding scheme Π that simulates any noiseless protocol π of length $s+2(r+1)$ with rate ρ_Π in the presence of up to δ_Π fraction of adversarial errors, and with running time $\mathcal{T}_\Pi$.*

Then for every $\gamma > 0$ there exists a deterministic interactive coding scheme Π' that simulates any noiseless protocol π' of length $n_\Lambda \cdot (s-2) \cdot (1-\gamma)$ with rate

$$\frac{\rho_\Lambda}{2 - \rho_\Lambda + 4/(s-2)} \cdot \rho_\Pi \cdot (1-\gamma),$$

and satisfies the following decoding guarantees:

1. *(BSC) Over $\mathsf{BSC}_{\delta_\Pi/2}$, the coding scheme Π' has failure probability*

$$\exp\left[-n_\Lambda\left(\frac{\delta_\Lambda}{36}\cdot\frac{s+2(r+1)}{\rho_\Pi}\cdot\frac{\delta_\Pi^2}{4}\cdot\gamma - H\left(\frac{\delta_\Lambda}{36}\cdot\gamma\right)\right)\right].$$

2. *(Adversarial channel) The coding scheme Π' is resilient to*

$$n_\Lambda \cdot \frac{\delta_\Lambda}{36} \cdot \frac{s+2(r+1)}{\rho_\pi} \cdot \delta_\Pi \cdot \gamma$$

adversarial errors.

Furthermore, the coding scheme Π' has running time $O(n_\Lambda \cdot (\mathcal{T}_\Lambda + \mathcal{T}_\Pi))$.

The following lemmas give the tree code that will be used as the 'outer code' in the concatenation step, and an exponential time deterministic coding scheme that will be used as the 'inner code' in the concatenation step.

LEMMA 3.2. (OUTER CODE) *There exists an absolute constant $\delta_0 > 0$ such that the following holds for every $\epsilon > 0$ and $n \in \mathbb{N}$. There exists a systematic $\mathbb{F}_2$-linear tree code $\Lambda : \Sigma_{\mathrm{in}}^n \to \Sigma_{\mathrm{out}}^n$ with $\Sigma_{\mathrm{in}} = \{0,1\}^{(\log n)/\epsilon}$, $\Sigma_{\mathrm{out}} = \{0,1\}^{(\log n)/\epsilon + \log n}$, rate $\frac{1}{1+\epsilon}$ and relative tree distance at least $\frac{\delta_0 \cdot \epsilon}{\log n}$. Furthermore, Λ can be encoded and decoded from up to $\frac{\delta_0 \cdot \epsilon}{2 \log n}$ fraction of errors in time* $\mathrm{poly}(n)$.

We prove the above lemma in Section 5.

LEMMA 3.3. (INNER CODE) *For every $\epsilon > 0$ there exists a deterministic interactive coding scheme Π that simulates any noiseless protocol π of length n with rate $1 - O\big(\sqrt{H(\epsilon)}\big)$ in the presence of up to ϵ fraction of adversarial errors. Furthermore, Π has running time* $\mathrm{poly}(n)$ *and can be constructed in time $2^{O(n)}$.*

We prove the above lemma in Section 6. We can now prove our main Theorems 1.1 and 1.2 based on the above Lemmas 3.1, 3.2 and 3.3.

Proof. [Proof of Theorem 1.1] Follows from Lemma 3.1 by letting Λ be the tree code guaranteed by Lemma 3.2 for constant ϵ and integer n_Λ such that $n = n_\Lambda((\log n_\Lambda)/\epsilon - 2)(1-\epsilon)$ (so $n_\Lambda = \Omega(\epsilon n/\log n)$), letting Π be the coding scheme guaranteed by Lemma 3.3 for constant 2ϵ, and letting $\gamma = \epsilon$. □

Proof. [Proof of Theorem 1.2] Follows from Lemma 3.1 by letting Λ be the tree code guaranteed by Lemma 3.2 for constant $\epsilon/16$ and integer n_Λ such that $n = n_\Lambda((\log n_\Lambda)/(\epsilon/16) - 2)(1-(\epsilon/16))$, letting Π be the coding scheme guaranteed by Lemma 3.3 for constant $H^{-1}(\epsilon^2/c)$ where c is a sufficiently large constant, and letting $\gamma = \epsilon/16$. □

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

4 The Concatenation Lemma: Proof of Lemma 3.1

We start with a high-level description of the coding scheme Π'. We describe below the coding scheme Π' for Alice, the coding scheme for Bob is symmetric.

Throughout the execution of the coding scheme Alice (respectively, Bob) maintains a string T^A that represents Alice's current guess for the transcript of the simulated protocol π' communicated so far. Alice also maintains a string $\hat{T}^B$ that represents Alice's current guess for the corresponding string T^B of Bob. When the execution of the coding scheme Π' is completed the outputs of Alice and Bob are T^A and T^B respectively.

The coding scheme Π' is executed for n_Λ iterations, where at iteration i Alice and Bob use the inner coding scheme Π to communicate the next block X_i of length $s-2$ of π' (assuming that the transcript of π' communicated so far is T^A and T^B, respectively), as well as a pair of *position strings* $p^A_{i-1}, p^B_{i-1} \in \{0,1\}^2$, and a pair of *hash strings* $h^A_{i-1}, h^B_{i-1} \in \{0,1\}^r$.

Alice (respectively, Bob) then performs, according to the output of the simulation via the inner coding scheme Π, one of three actions: she either appends her noisy version X^A_i of X_i to T^A, or she leaves T^A unchanged, or she erases the last block of length $s-2$ from T^A. She then records her action in the i-th position string p^A_i (since there are only three possible actions those could be recorded using 2 bits).

Lastly, Alice views the string $(\sigma_{\text{in}})^A_i := (p^A_i, X^A_i) \in \{0,1\}^s$ as the systematic part of the i-th output symbol of the tree code Λ and lets $(\sigma_{\text{out}})^A_i$ be the corresponding i-th output symbol of the tree code. The i-th hash string $h^A_i \in \{0,1\}^r$ is set to be the redundant part of $(\sigma_{\text{out}})^A_i$. As described above, both the strings p^A_i and h^A_i will be communicated by Alice in iteration $i+1$. Note that for every i, the string $((\sigma_{\text{in}})^A_1, \ldots, (\sigma_{\text{in}})^A_i)$ records all the actions of Alice on T^A till iteration i and so, if decoded correctly by Bob, then Bob can extract the value of T^A at iteration i from this string (same goes for Alice). The formal definition of the coding scheme Π' appears below.

4.1 The coding scheme Π'

Coding scheme $(\Pi')^A$ for Alice:
Initialize: $T^A := \emptyset$, $\hat{T}^B := \emptyset$.
For $i = 1, \ldots, n_\Lambda$ iterations:

1. Recall that p^A_{i-1} denotes the first 2 bits of $(\sigma_{\text{out}})^A_{i-1}$ and let h^A_{i-1} denote the last r bits of $(\sigma_{\text{out}})^A_{i-1}$ (for $i=1$ let $(\sigma_{\text{out}})^A_0 := 0^{s+r}$).
2. Simulate the protocol $\pi^A\big(|T^A|, (T^A, 0^{n_\Lambda(s-2)-|T^A|}), p^A_{i-1}, h^A_{i-1}\big)$ below using the inner coding scheme Π. Let the sequence $(p^A_{i-1}, \hat{p}^B_{i-1}, h^A_{i-1}, \hat{h}^B_{i-1}, X^A_i)$ denote the output of the simulation where $p^A_{i-1}, \hat{p}^B_{i-1} \in \{0,1\}^2$, $h^A_{i-1}, \hat{h}^B_{i-1} \in \{0,1\}^r$ and $X^A_i \in \{0,1\}^{s-2}$.
3. Let $(\widehat{\sigma}_{\text{out}})^B_{i-1} := \big(\hat{p}^B_{i-1}, X^A_{i-1}, \hat{h}^B_{i-1}\big)$. Decode the sequence $((\widehat{\sigma}_{\text{out}})^B_1, \ldots, (\widehat{\sigma}_{\text{out}})^B_{i-1})$ using the decoding algorithm for Λ. Let $((\widehat{\sigma}_{\text{in}})^B_1, \ldots, (\widehat{\sigma}_{\text{in}})^B_{i-1})$ be the decoded message and let $\hat{T}^B$ be the transcript represented by this string (if $i=1$ then set $\hat{T}_B = \emptyset$).
4. If $T^A = \hat{T}^B$ append X^A_i to T^A and set $p^A_i := 01$.
5. Otherwise, if $T^A \neq \hat{T}^B$ and $|T^A| < |\hat{T}^B|$ set $p^A_i := 00$.
6. Otherwise, if $T^A \neq \hat{T}^B$ and $|T^A| \geq |\hat{T}^B|$ erase last $s-2$ bits from T^A and set $p^A_i := 10$.
7. Let $(\sigma_{\text{in}})^A_i := (p^A_i, X^A_i)$ and let $(\sigma_{\text{out}})^A_i$ be the i-th symbol of $\Lambda((\sigma_{\text{in}})^A_1, \ldots, (\sigma_{\text{in}})^A_i)$. Note that since Λ is systematic it holds that $(\sigma_{\text{in}})^A_i$ is a prefix of $(\sigma_{\text{out}})^A_i$.

The output of the coding scheme is the prefix of T^A of length $n_\Lambda \cdot (s-2) \cdot (1-\gamma)$.

Next we describe the protocol π. This protocol is simulated by the inner coding scheme Π at Step 2 of the coding scheme Π'. The protocol π receives as input an integer $1 \leq t \leq n_\Lambda(s-2)$, a transcript string $T \in \{0,1\}^{n_\Lambda(s-2)}$, a position string $p \in \{0,1\}^2$ and a hash string $h \in \{0,1\}^r$. The description of π for Alice's side, denoted π^A, is the following.

Protocol $\pi^A(t, T, p, h)$ for Alice:

1. Send p, h and receive $\hat{p}$, $\hat{h}$ (this is done bit by bit).
2. Communicate bits $[t+1, \ldots, t+(s-2)]$ of the protocol π' assuming that the first t bits of π' communicated so far are the first t bits of T.

4.2 Analysis

4.2.1 Rate and running time The coding scheme Π' runs for n_Λ iterations and at each iteration the number of bits communicated is $((s-2)+2(r+2))/\rho_\Pi$.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

Consequently, the rate of the coding scheme Π' is

$$\begin{aligned}\frac{|\pi'|}{|\Pi'|} &= \frac{n_\Lambda \cdot (s-2) \cdot (1-\gamma)}{n_\Lambda \cdot ((s-2)+2(r+2))/\rho_\Pi} \\ &= \frac{s-2}{2(s+r)-(s-2)} \cdot \rho_\Pi \cdot (1-\gamma) \\ &= \frac{s-2}{2(s-2)/\rho_\Lambda + 4/\rho_\Lambda - (s-2)} \cdot \rho_\Pi \cdot (1-\gamma) \\ &= \frac{\rho_\Lambda}{2+4/(s-2)-\rho_\Lambda} \cdot \rho_\Pi \cdot (1-\gamma).\end{aligned}$$

To analyze the running time note that the running time of each iteration is $O(\mathcal{T}_\Lambda + \mathcal{T}_\Pi)$ and therefore the total running time is $O(n_\Lambda \cdot (\mathcal{T}_\Lambda + \mathcal{T}_\Pi))$.

4.2.2 Decoding guarantees To analyze the decoding guarantees we define a potential function Φ as follows. Let t^+ be the number of blocks of length $s-2$ contained in the longest prefix on which T^A and T^B agree, and let $t^- = \frac{|T^A|+|T^B|}{s-2} - 2t^+$. Let $\Phi = t^+ - t^-$. Note that the simulation succeeds if $\Phi \geq n_\Lambda \cdot (1-\gamma)$ when the execution of Π' is completed since in this case it holds that $t^+ \geq n_\Lambda \cdot (1-\gamma)$, and so T^A and T^B both contain the correct length $n_\Lambda \cdot (s-2) \cdot (1-\gamma)$ transcript of the protocol π' as a prefix.

To bound the potential we shall use the notion of a *good iteration*. We say that an iteration i is *good* if the following pair of conditions hold:

1. At Step 2 of iteration i, the simulation of π via the inner coding scheme Π is successful.
2. At Step 3 of iteration i, it holds that $T^A = \hat{T}^A$ and $T^B = \hat{T}^B$.

The following claim says that the potential increases by at least 1 at a good iteration and decreases by at most 3 otherwise.

CLAIM 4.1. *The potential decreases by at most* 3 *at the end of each iteration. Furthermore, at the end of a good iteration the potential increases by at least* 1.

The above claim implies that the simulation of π' via Π' succeeds as long as the number of bad iterations throughout the execution of Π' is at most $n_\Lambda\gamma/4$. The following claim shows that to bound the number of bad iterations it suffices to bound the number of iterations in which the first condition in the definition of a good iteration does not hold.

CLAIM 4.2. *If the first condition in the definition of a good iteration does not hold in at most m iterations, then the number of bad iterations is at most $9m/\delta_\Lambda$.*

Before proving the above claim we show how the required decoding guarantees follow from it. By the above claim the simulation of Π' is successful as long as the number of iterations in which the simulation at Step 2 failed is at most $\delta_\Lambda n_\Lambda \gamma/36$. Over $\mathsf{BSC}_{\delta_\Pi/2}$, since the inner coding scheme Π can handle δ_Π fraction of adversarial errors, the probability that the simulation at Step 2 fails is at most

$$\exp\left(-\left(\frac{\delta_\Pi}{2}\right)^2 \cdot \frac{s+2(r+1)}{\rho_\Pi}\right),$$

independently for each iteration. Therefore the probability of having more than $\delta_\Lambda n_\Lambda \gamma/36$ iterations in which the simulation at Step 2 fails is at most

$$\begin{aligned}&\binom{n_\Lambda}{\delta_\Lambda n_\Lambda \gamma/36} \exp\left(-\frac{\delta_\Pi^2}{4} \cdot \frac{s+2(r+1)}{\rho_\Pi} \cdot \frac{\delta_\Lambda n_\Lambda \gamma}{36}\right) \\ &= \exp\left[-n_\Lambda \left(\frac{\delta_\Lambda}{36} \cdot \frac{s+2(r+1)}{\rho_\Pi} \cdot \frac{\delta_\Pi^2}{4} \cdot \gamma - H\left(\frac{\delta_\Lambda}{36} \cdot \gamma\right)\right)\right].\end{aligned}$$

Similarly, over an adversarial channel the simulation at Step 2 is successful as long as the number of errors in the simulation is at most

$$\delta_\Pi \cdot \frac{s+2(r+1)}{\rho_\Pi},$$

and so the simulation of the coding scheme Π' is successful as long as the total number of errors is at most

$$n_\Lambda \cdot \frac{\delta_\Lambda}{36} \cdot \frac{s+2(r+1)}{\rho_\pi} \cdot \delta_\Pi \cdot \gamma.$$

It remains to prove Claim 4.2.

Proof. [Proof of Claim 4.2] By symmetry, it suffices to show that there are at most $4m/\delta_\Lambda$ iterations in which $T_B \neq \hat{T}_B$ at Step 3.

Fix an iteration $i+1$ in which $T_B \neq \hat{T}_B$ at Step 3 and let $((\widehat{\sigma}_{\text{in}})_1^B, \ldots, (\widehat{\sigma}_{\text{in}})_i^B)$ be the decoded message at this step. By the decoding guarantee of Λ there exists $t(i) \in [i]$ such that in at least δ_Λ fraction of the iterations $j \in [t(i), i]$ the simulation at Step 2 failed in either iteration j or iteration $j+1$ (since X_j is transmitted on iteration j but p_j and h_j are transmitted only on iteration $j+1$). This implies in turn that in at least $\delta_\Lambda/2$ fraction of the iterations $j \in [t(i), i+1]$ the simulation at Step 2 failed in iteration j.

Let

$$\mathcal{I} = \left\{[t(i), i+1] \mid T_B \neq \hat{T}_B \text{ at Step 3 of iteration } i+1\right\}.$$

Since for each iteration $i+1$ in which $T^B \neq \hat{T}^B$ it holds that $i+1 \in \bigcup \mathcal{I} = \bigcup_{I \in \mathcal{I}} I$, it suffices to show that $|\bigcup \mathcal{I}| \leq 4m/\delta_\Lambda$. Lemma 7 in [21] says that there

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

exists a subset $\mathcal{I}' \subseteq \mathcal{I}$ of distjoint intervals such that $|\bigcup \mathcal{I}'| \geq |\bigcup \mathcal{I}|/2$. The proof is completed by noting that our assumption that the simulation at Step 2 failed in at most m iterations implies that $|\bigcup \mathcal{I}'| \leq 2m/\delta_\Lambda$, and so $|\bigcup \mathcal{I}| \leq 4m/\delta_\Lambda$. □

5 The Outer Code: Proof of Lemma 3.2

5.1 The tree code construction

The main ingredient in our tree code construction is the following lemma which shows the existence of a *systematic* error-correcting code $C : \Sigma_{\text{in}}^k \to \Sigma_{\text{out}}^k$ with appropriate parameters. Specifically, this lemma shows for every integers k, n which satisfy that $\Omega((\log n)/\epsilon) \leq k \leq n$ the existence of a systematic code $C : \Sigma_{\text{in}}^k \to \Sigma_{\text{out}}^k$ with $|\Sigma_{\text{in}}| = \text{poly}(n)$, $|\Sigma_{\text{out}}| = \text{poly}(n)$, rate $1 - O(\frac{\epsilon}{\log n})$ and relative distance $\Omega(\frac{\epsilon}{\log n})$. The lemma follows by an application of Lemmas 2.1 and 2.2, and we defer its proof to Section 5.4.

LEMMA 5.1. *There exists an absolute constant $k_0 \in \mathbb{N}$ such that the following holds for every $\epsilon > 0$ and integers $k, n \in \mathbb{N}$ such that $k_0 \cdot (\log n)/\epsilon \leq k \leq n$. There exists a systematic $\mathbb{F}_2$-linear code $C : \Sigma_{\text{in}}^k \to \Sigma_{\text{out}}^k$ with $\Sigma_{\text{in}} = \{0,1\}^{(\log n)/\epsilon}$, $\Sigma_{\text{out}} = \{0,1\}^{(\log n)/\epsilon+1}$, rate $\rho' := \frac{1}{1+\epsilon/\log n}$ and relative distance at least $\delta' := \frac{1}{2(\log n)/\epsilon+1}$. Furthermore, C can be encoded and decoded from up to $\delta'/2$ fraction of errors in time* $\text{poly}(n)$.

The tree code Λ is constructed as follows. Let $m := k_0 \cdot (\log n)/\epsilon$, for simplicity assume that both m and n are powers of 2. The encoding $\Lambda(x)$ of a message $x \in \Sigma_{\text{in}}^n$ will be the pointwise concatenation of the message string x with $\log n - \log m + 1$ binary strings $x^{(\log m)}, \ldots, x^{(\log n)} \in \{0,1\}^n$, where for $\log m \leq t \leq \log n$ the string $x^{(t)} \in \{0,1\}^n$ is defined as follows. Let $C^{(t)} : \Sigma_{\text{in}}^{2^t} \to \Sigma_{\text{out}}^{2^t}$ be the systematic code given by the above lemma for constant ϵ and message length $k = 2^t$, and let $R^{(t)} : \Sigma_{\text{in}}^{2^t} \to \{0,1\}^{2^t}$ be the redundant part of $C^{(t)}$. Divide the string x into $n/2^t$ blocks $z_1, \ldots, z_{n/2^t}$ of length 2^t each, and let $x^{(t)} = (0^{2^t}, R^{(t)}(z_1), \ldots, R^{(t)}(z_{n/2^t-1}))$. See Figure 2.

We clearly have that Λ can be encoded in time $\text{poly}(n)$. Note furthermore that Λ is systematic and $\mathbb{F}_2$-linear and that the input alphabet size of Λ is $2^{\log n/\epsilon}$ and the output alphabet size of Λ is $2^{\log n/\epsilon} \cdot 2^{\log n - \log m + 1} \leq 2^{\log n/\epsilon + \log n}$. The rate of Λ is at least

$$\frac{(\log n)/\epsilon}{(\log n)/\epsilon + \log n} = \frac{1}{1+\epsilon}.$$

It remains to analyze the distance and decoding guarantee of Λ.

5.2 Distance

The distance guarantee stems from the fact that as long as we look at two different messages x, y that differ in their suffixes of length $\geq 2m$, then the encoding at these suffixes completely includes a pair of codewords $C^{(t)}(x') \neq C^{(t)}(y')$ for some $\log m \leq t \leq \log n$. Below, we show that either the suffix is shorter than $2m$ and then the required distance trivially holds, or we find the maximal value of t for which the above holds and then the required distance follows from the distance guarantee of the code $C^{(t)}$.

Let $x, y \in \Sigma_{\text{in}}^n$ be a pair of distinct messages and let $i \in [n]$ be the first coordinate on which x and y differ. Fix an index j such that $i \leq j \leq n$, we shall show below that

$$\begin{aligned}
&\text{dist}\Big(\Lambda(x)[i,j], \Lambda(y)[i,j]\Big) \\
&\geq \max\left\{\frac{\epsilon}{2k_0 \log n}, \frac{\delta'}{8}\right\} \\
&= \max\left\{\frac{\epsilon}{2k_0 \log n}, \frac{1}{16(\log n)/\epsilon + 8}\right\},
\end{aligned}$$

and so Lemma 3.2 holds with $\delta_0 := 1/(32k_0)$.

If $j - i < 2m$ then we have that

$$\begin{aligned}
\text{dist}\Big(\Lambda(x)[i,j], \Lambda(y)[i,j]\Big) &\geq \frac{1}{j-i+1} \\
&\geq \frac{1}{2m} \\
&= \frac{\epsilon}{2k_0 \log n},
\end{aligned}$$

where the first inequality follows since $(\Lambda(x))_i \neq (\Lambda(y))_i$ due to our assumption that $x_i \neq y_i$ and the tree code being systematic.

Next assume that $j - i \geq 2m$. Let t be maximal integer such that $2 \cdot 2^t \leq j - i$ and let $i_0 := \left\lfloor \frac{i-1}{2^t} \right\rfloor \cdot 2^t$ be $i-1$ rounded down to the nearest multiple of 2^t. Note that

$$i_0 + 1 \leq i < i_0 + 1 + 2^t < i_0 + 2 \cdot 2^t \leq j,$$

and

$$j - i < 4 \cdot 2^t$$

due to maximality of t.

To show the required distance note that the systematic part $x[i_0+1, i_0+2^t]$ of $C^{(t)}(x[i_0+1, i_0+2^t])$ is included in $\Lambda(x)[i_0+1, i_0+2^t]$ due to Λ being systematic, and the redundant part $R^{(t)}(x[i_0+1, i_0+2^t])$ of $C^{(t)}(x[i_0+1, i_0+2^t])$ is included in $\Lambda(x)[i_0+1+2^t, i_0+2\cdot 2^t]$ by construction of Λ, and that the same holds for y.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

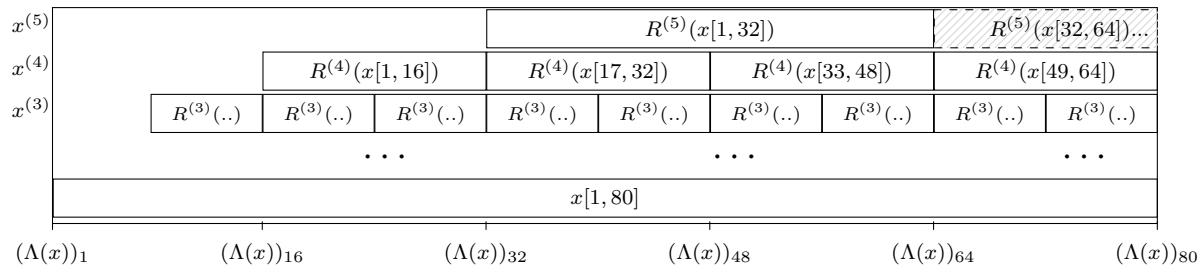

Figure 2: An illustration of the first 80 indices of $\Lambda(x)$, the encoding of $x \in \Sigma_{\text{in}}^n$ using our tree code.

Furthermore, the assumption that $x_i \neq y_i$ implies that $x[i_0+1, i_0+2^t] \neq y[i_0+1, i_0+2^t]$ and so by the distance guarantee of $C^{(t)}$ we have that

$$\text{dist}\Big(C^{(t)}\big(x[i_0+1, i_0+2^t]\big), C^{(t)}\big(y[i_0+1, i_0+2^t]\big)\Big) \geq \delta'.$$

This implies in turn that either

$$\text{dist}\Big(x[i_0+1, i_0+2^t], y[i_0+1, i_0+2^t]\Big) \geq \frac{\delta'}{2}$$

or

$$\text{dist}\Big(R^{(t)}\big(x[i_0+1, i_0+2^t]\big), R^{(t)}\big(y[i_0+1, i_0+2^t]\big)\Big) \geq \frac{\delta'}{2}.$$

Finally, note that in either case it follows that

$$\begin{aligned}\text{dist}\Big(\Lambda(x)[i,j], \Lambda(y)[i,j]\Big) &\geq \frac{(\delta'/2)\cdot 2^t}{j-i+1} \\ &\geq \frac{(\delta'/2)\cdot 2^t}{4\cdot 2^t} \\ &= \frac{\delta'}{8},\end{aligned}$$

where the first inequality is due to the fact that $i_0+1 \leq i < i_0+1+2^t < i_0+2\cdot 2^t \leq j$ and i is the first coordinate on which x and y differ, and the second inequality is due to the fact that $j-i < 4\cdot 2^t$.

5.3 Decoding

Recall that the decoding procedure is given a word $w \in \Sigma_{\text{out}}^j$ for some $1 \leq j \leq n$ and is required to output a vector $y \in \Sigma_{\text{in}}^j$ such that $y = x$ whenever $x \in \Sigma_{\text{in}}^j$ is such that $\text{dist}_{\text{sfx}}(\Lambda(x), w) \leq \frac{\delta_0 \cdot \epsilon}{2\log n}$.

For a given word $w \in \Sigma_{\text{out}}^j$, the decoded word $y \in \Sigma_{\text{in}}^j$ is obtained as follows. We decode w in parts according to its partitioning into blocks corresponding to the codes $C^{(t)}$. Specifically, we start from the largest t so that a codeword of $C^{(t)}$ is fully included in the prefix of w. We then move on to decode the remaining suffix in an iterative manner. We proceed this way until the interval at hand is shorter than $2m$, in which case we assume that no errors occurred in the interval.

The formal description of the decoding procedure follows.

Decoding procedure on input $w \in \Sigma_{out}^j$:

0. $\ell := 1$ // Left index of current interval
1. If $j - \ell < 2m$, set $y[\ell, j]$ to be the systematic part of $w[\ell, j]$ and output y.
2. Otherwise, let t be maximal integer such that $2 \cdot 2^t \leq j - \ell$.
3. Decode the part of $w[\ell, \ell - 1 + 2 \cdot 2^t]$ that corresponds to the encoding of the code $C^{(t)}$ using the decoding procedure for $C^{(t)}$, and set $y[\ell, \ell - 1 + 2^t]$ to be the result of the decoding.
4. Set $\ell := \ell + 2^t$ and return to Step 1.

Let us give an example of the decoding process of $w \in \Sigma_{\text{out}}^{75}$. (Recall Figure 2.) For this example, let us assume that $m = 8 = 2^3$. We begin by decoding $y[1, 32]$; this is done by decoding the code $C^{(5)}$ whose systematic part lies in $w[1, 32]$ and redundant part $R^{(5)}(x[1, 32])$ lies in $w[33, 64]$. Note that we could not use the code $C^{(6)}$ since its redundant part would be in the interval $[65, 128]$ which is beyond the range of w. After we set $y[1, 32]$, we move on to the next interval. We cannot decode $y[33, 64]$ using the next $C^{(5)}$ since its redundant part lies beyond the range of w, and we need to reduce the scale to $t = 4$. Hence, the next part we decode is $y[33, 48]$, which is obtained using the code $C^{(4)}$ whose systematic part lies in $w[33, 48]$ and redundant part $R^{(4)}(x[33, 48])$ lies in $w[49, 64]$. The next $C^{(4)}$ is again beyond the currently decoded w and we reduce the scale to $t = 3$. Using the code $C^{(3)}$ we decode $y[49, 56]$, and also $y[57, 64]$. Finally, we are left with the interval $[65, 75]$ whose length is $11 < 2m$; we assume that there

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

are no errors in this interval and simply set $y[65,75]$ to be the systematic part of $w[65,75]$.

We clearly have that the decoding procedure runs in time $\mathrm{poly}(n)$. To show that the decoding procedure satisfies the required decoding guarantee we observe that our assumption—that the distance of w from $\Lambda(x)$ is small on every suffix—implies that at each iteration the part of $w[\ell, \ell-1+2\cdot 2^t]$ that corresponds to the encoding of $C^{(t)}$ is close to $C^{(t)}(x[\ell, \ell-1+2^t])$. Consequently, the decoding guarantee of $C^{(t)}$ implies that $y[\ell, \ell-1+2^t] = x[\ell, \ell-1+2^t]$ for every iteration in which $j-\ell \geq 2m$.

In more detail, suppose that $x \in \Sigma_{\mathrm{in}}^j$ is such that $\mathrm{dist}_{\mathrm{sfx}}(\Lambda(x), w) \leq \frac{\delta_0\cdot\epsilon}{2\log n}$. We shall show that at each iteration the coordinates of y are set to the corresponding coordinates of x and so $y = x$.

If $j-\ell < 2m$ at some iteration then we have that

$$\mathrm{dist}\Big(\Lambda(x)[\ell,j], w[\ell,j]\Big) \leq \frac{\delta_0\cdot\epsilon}{2\log n} = \frac{1}{64m} < \frac{1}{j-\ell+1},$$

where the equality follows due to our choice of $m = k_0(\log n)/\epsilon$ and $\delta_0 = 1/(32k_0)$. This implies in turn that $w[\ell,j] = \Lambda(x)[\ell,j]$ and so the systematic part of $w[\ell,j]$ equals $x[\ell,j]$ and consequently $y[\ell,j] = x[\ell,j]$.

Next assume that $j-\ell \geq 2m$. To show the required decoding guarantee in this case note that our assumption implies that

$$\mathrm{dist}\Big(\Lambda(x)[\ell,j], w[\ell,j]\Big) \leq \frac{\delta_0\cdot\epsilon}{2\log n}.$$

Furthermore, due to maximality of t we have that $j-\ell < 4\cdot 2^t$, and consequently it holds that

$$\begin{aligned}
&\mathrm{dist}\Big(\Lambda(x)[\ell,\ell-1+2^t], w[\ell,\ell-1+2^t]\Big)\\
&\leq \frac{4\cdot 2^t\cdot(\delta_0\cdot\epsilon)/(2\log n)}{2^t}\\
&= \frac{2\cdot\delta_0\cdot\epsilon}{\log n}\\
&\leq \frac{\delta'}{4},
\end{aligned}$$

and similarly

$$\begin{aligned}
&\mathrm{dist}\Big(\Lambda(x)[\ell+2^t,\ell-1+2\cdot 2^t], w[\ell+2^t,\ell-1+2\cdot 2^t]\Big)\\
&\leq \frac{\delta'}{4}.
\end{aligned}$$

This implies in turn that the part of $w[\ell,\ell-1+2\cdot 2^t]$ that corresponds to the encoding of $C^{(t)}$ is of relative distance at most $\delta'/2$ from $C^{(t)}(x[\ell,\ell-1+2^t])$, and so by the decoding guarantee of $C^{(t)}$ it holds that $y[\ell,\ell-1+2^t] = x[\ell,\ell-1+2^t]$.

5.4 Proof of Lemma 5.1

We now complete the proof of Lemma 3.2 by proving Lemma 5.1. Lemma 5.1 follows by substituting $\rho = 1/2$, $s = (\log n)/\epsilon$ and $r = 1$ in the following lemma which shows the existence of a systematic error-correcting code with good rate and distance.

LEMMA 5.2. *For every $0 < \rho < 1$ there exist $\delta > 0$ and integer $k_0 \in \mathbb{N}$ such that the following holds for any integers $k, s, r \in \mathbb{N}$ which satisfy that $k\cdot\frac{\rho r}{s} \geq k_0$ and $s \geq \log(k(1+\frac{\rho r}{s}))$. There exists a systematic $\mathbb{F}_2$-linear code $C : \Sigma_{\mathrm{in}}^k \to \Sigma_{\mathrm{out}}^k$ with $\Sigma_{\mathrm{in}} = \{0,1\}^s$, $\Sigma_{\mathrm{out}} = \{0,1\}^{s+r}$, rate $\frac{s}{s+r}$ and relative distance at least $\delta' := \min\left\{\delta, 1-\frac{s/\rho}{s/\rho+r}\right\}$. Furthermore, C can be encoded and decoded from up to $\delta'/2$ fraction of errors in time $\mathrm{poly}(k,s,r)$.*

Proof. Since C is systematic it suffices to define the redundant part R of C. Roughly speaking, $R(x)$ is obtained by first encoding the message x via a systematic Reed-Solomon code, then encoding the redundant part of the resulting codeword with an asymptotically good binary code, and finally spreading the resulting bits evenly between the k coordinates of $R(x)$.

Formally, let δ and k_0 be the constants guaranteed by Lemma 2.2 for rate ρ, and let B be the asymptotically good binary code guaranteed by this fact for rate ρ and message length $k\cdot\frac{\rho r}{s}$ (recall that we assumed that $k\cdot\frac{\rho r}{s} \geq k_0$). Let RS be the Reed-Solomon code guaranteed by Lemma 2.1 for message length k and block length $k(1+\frac{\rho r}{s})$ over a field $\mathbb{F}$ of size 2^s, and note that our assumptions imply that $2^s \geq k(1+\frac{\rho r}{s})$. By performing Guassian elimination, we may assume without loss of generality that the code RS is systematic, that is, for every $x \in \mathbb{F}^k$ it holds that $\mathrm{RS}(x) = (x, R'(x))$ for some string $R'(x) \in \mathbb{F}^{k\rho r/s}$.

Next we define the redundant part R of C. To this end, fix a string $x \in \Sigma_{\mathrm{in}}^k = \mathbb{F}^k$ and let $R'(x) \in \mathbb{F}^{k\rho r/s}$ be the redundant part of the encoding of x via the Reed-Solomon code RS. Next view $R'(x)$ as a binary string in $\{0,1\}^{k\rho r}$ via the usual $\mathbb{F}_2$-linear isomorphism and encode this binary string via the asymptotically good binary code B, let $z_x \in \{0,1\}^{kr}$ denote the resulting string. Finally, divide the string z_x into k blocks of size r, and for every $1 \leq i \leq k$ let $(R(x))_i \in \{0,1\}^r$ be the i-th block of z_x.

Nest we analyze the properties of C. It can be verified that C has the required rate $\frac{s}{s+r}$. To see that the relative distance of C is at least δ', let $x \neq y \in \Sigma_{\mathrm{in}}^k$ be a pair of strings. If

$$\mathrm{dist}(x,y) \geq 1 - \frac{k}{k(1+\rho r/s)} = 1 - \frac{s/\rho}{s/\rho+r}$$

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

then we are done due to C being systematic. Otherwise, due to the distance guarantee of the code RS we must have that $R'(x) \neq R'(y)$, and consequently the distance guarantee of the code B implies that $\mathrm{dist}(z_x, z_y) \geq \delta$. Finally, note that grouping the coordinates of z_x and z_y cannot decrease the relative distance between the pair of strings, and so we must have that $\mathrm{dist}(R(x), R(y)) \geq \delta$ as well. The decoding guarantees of C follow from similar considerations, based on the decoding guarantees of the codes RS and B. □

6 The Inner Code: Proof of Lemma 3.3

In this section we outline our inner code and the proof of Lemma 3.3. Detailed proofs and analysis appear in the full version of this paper

The inner code is obtained via a derandomization of a randomized interactive coding scheme due to Haeupler [14, Algorithm 3]. The main use of randomness in [14] is to allow the parties to check, with high probability, whether or not they are synchronized (e.g., hold the same partial transcript). To this end, each party chooses a random hash function and communicates a short hash of its own state. Note that due to this shrinkage in length, it may happen that although the parties are unsynchronized, the hash values they exchange are the same, leading the parties to falsely believe they are synchronized. Such an event is called a *hash collision*. We show how to devise a *deterministic* variant of the coding scheme of [14], in which we fix the randomness, and show that there exists a fixing that is "good" for all possible runs, namely, the amount of hash collisions that can occur for that fixing is low enough to complete the simulation correctly.

To this end, we observe that when the adversary is limited to ϵ fraction of errors there are only $2^{O(H(\epsilon)n)} = 2^{O(\log(1/\epsilon)\epsilon n)}$ different noise patterns that should be considered; denote these as *typical noise patterns*. We then carefully modify the way the coding scheme of [14] compares the states the parties hold, using linear hash functions. The linearity of the hash functions along with the specific way in which we perform the comparisons make hash collisions depend (roughly) only on the specific noise pattern and the randomness string, and most importantly, (almost) independent of the specific noiseless protocol π that is simulated by the coding scheme and the inputs (x, y) of the parties (The fact that hash collisions do not entirely depend only on the noise pattern and the randomness string creates further complications in our proof which we ignore in the discussion below).

Finally, we show that if we increase the output length of the hash functions from a constant (as used in [14]) to $c' \log(1/\epsilon)$ for some constant c', then for each typical noise pattern, at least a $1 - 2^{-\Omega(c' \log(1/\epsilon)\epsilon n)}$ fraction of the randomness strings lead to at most ϵn hash collisions which is a small enough number of hash collisions that allows the simulation to be completed correctly. By setting c' to be a large enough constant, a union bound proves that there must exist a single randomness string that is "good" for *all* possible typical noise patterns. A by-product of the increase in the output length of the hash functions is that the rate of the coding scheme slightly reduces from $1 - O(\sqrt{\epsilon})$ to $1 - O(\sqrt{H(\epsilon)})$.

Concretely, we prove Lemma 3.3 in two steps. In the first step we slightly modify the original scheme of [14], specifically, by carefully modifying the way the hash comparisons are performed and slightly increasing the output length of the hash functions, as outlined above. In the second step we derandomize this latter modified coding scheme.

The two steps are given in Sections 6.1 and 6.2 below, respectively. The analysis below builds on the analysis of [14] and is not self-contained. We refer the reader to the longer version of that paper [15], and in the following all the references (lemma numbers, line numbers, variable names, etc.) correspond to that version.

6.1 Modified scheme

We now show how to modify the randomized coding scheme given by Algorithm 3 in [15] to obtain a randomized coding scheme $\widetilde{\Pi}$ that is more suitable for derandomization, and state some properties of the modified scheme $\widetilde{\Pi}$ that we shall use for the derandomization step. We start by describing the modified coding scheme $\widetilde{\Pi}$.

Let $\widetilde{\Pi}$ be the coding scheme that is obtained from Algorithm 3 in [15] via the following modifications.

1. (Output length of hash functions) The output length o of the hash functions is increased from $\Theta(1)$ to $c' \cdot \log(1/\epsilon)$ for sufficiently large constant c' to be determined later on. Consequently, the number r_c of check bits per iteration is also increased from $\Theta(1)$ to $\Theta(\log(1/\epsilon))$, the length of the blocks r is increased from $\Theta\big(\sqrt{1/\epsilon}\big)$ to $\Theta\big(\sqrt{\log(1/\epsilon)/\epsilon}\big)$, and the number of iterations R_{total} is decreased to
$$\lceil n/r + 65\epsilon n \rceil = \Theta\big(\sqrt{\epsilon/\log(1/\epsilon)}\big)n + 65\epsilon n.$$
2. (Seed length) Our modified hash comparisons described below apply hash functions to strings of length $\Theta(n \log n)$, as opposed to length $\Theta(n)$ as is done in Algorithm 3 in [15]. To this end, we increase the seed length s of the hash functions per iteration from $\Theta(n)$ to $\Theta(n \log n)$. Note that in

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

this case the random string R at Line 5 can still be obtained by exchanging $\Theta\big(\sqrt{\epsilon\log(1/\epsilon)}\big)n$ random bits sampled from a δ-biased distribution with bias $\delta = 2^{-\Theta(no/r)} = 2^{-\Theta\left(\sqrt{\epsilon\log(1/\epsilon)}\right)n}$.

3. (Position string $N(T)$) To make hash collisions depend (roughly) only on the noise pattern and the randomness string, the parties maintain throughout the execution of the coding scheme $\widetilde{\Pi}$ a *position string* $N(T) \in [R_{\text{total}}]^{R_{\text{total}}\cdot r}$ whose i-th coordinate equals the iteration in which the i-th bit of T was added to T, or is empty in the case in which the i-th bit of T is empty. We denote by $N'(T) \in \{0,1\}^{R_{\text{total}}\cdot r\cdot\log(R_{\text{total}})}$ the binary string obtained from $N(T)$ by replacing each coordinate of $N(T)$ with its binary representation of length $\log(R_{\text{total}})$ (we add zeros to the left of the binary representation if its length is shorter than $\log(R_{\text{total}})$).

4. (Hash comparisons) Roughly speaking, our new hash comparisons will apply an $\mathbb{F}_2$-linear hash function (specifically, the inner product function) to both the transcript T and the vector $N'(T)$ that encodes the iterations in which each of the bits in T were added to T. Specifically, in Line 9 we replace $\text{hash}_S(k)$, $\text{hash}_S(T)$, $\text{hash}_S(T[1,\text{MP1}])$, $\text{hash}_S(T[1,\text{MP2}])$ with $hash_S(k)$, $\widetilde{\text{hash}}_S\big(T\big)$, $\widetilde{\text{hash}}_S\big(T[1,\text{MP1}]\big)$, $\widetilde{\text{hash}}_S\big(T[1,\text{MP2}]\big)$, where the function hash_S is as defined in [15] and the function $\widetilde{\text{hash}}_S$ is defined as follows.

   For integers m, o and a seed $S \in \{0,1\}^{m\cdot o}$ let $h_S : \{0,1\}^{\le m} \to \{0,1\}^o$ be the $\mathbb{F}_2$-linear hash function that satisfies, for every $x \in \{0,1\}^{\le m}$ and $i \in [o]$, that

$$\big(h_S(x)\big)_i = \big\langle x, S[(i-1)\cdot m + 1, im]\big\rangle,$$

   where $\langle a,b\rangle = \sum_{i=1}^{m} a_i\cdot b_i \pmod 2$ is the inner product mod 2 of $a,b \in \{0,1\}^m$ (if $|x| < m$ then we assume that x is padded with zeroes to the right up to length m). We note that for every seed S the function h_S is $\mathbb{F}_2$-linear, i.e., for any two strings $x, y \in \{0,1\}^m$ it holds that $h_S(x \oplus y) = h_S(x) \oplus h_S(y)$.

   Finally, for $m = R_{\text{total}}\cdot r\cdot\log(R_{\text{total}}) = \Theta(n\log n)$, $o = c'\log(1/\epsilon)$ and a seed $S \in \{0,1\}^{m\cdot o}$ we let

$$\widetilde{\text{hash}}_S : \{0,1\}^{\le R_{\text{total}}\cdot r} \to \{0,1\}^{3o}$$

   be the hash function which satisfies that

$$\widetilde{\text{hash}}_S(x) = \Big(h_S(x), h_S(|x|), h_S(N'(x))\Big)$$

   for every string $x \in \{0,1\}^{\le R_{\text{total}}\cdot r}$.

6.2 Derandomization

In order to derandomize the coding scheme $\widetilde{\Pi}$ defined above we proceed according to the program outlined at the beginning of this section. Specifically, we observe that as long as each block in T_A was added at the same iteration in which the corresponding block in T_B was added (that is, $N(T_A) = N(T_B)$) then T_A and T_B differ only by the noise pattern corresponding to the iterations in which the blocks in T_A and T_B were added. Since the hash function h_S we use is $\mathbb{F}_2$-linear, in this case we have that hash collisions, when comparing T_A and T_B, depend only on the noise pattern and the seed S used in these iterations. However, when $N(T_A) \neq N(T_B)$, hash collisions may not depend entirely on the noise pattern and the random seed, and this creates further complications in our proof.

To cope with the above situation we replace in our analysis *noise patterns* with *behavior patterns* which include the noise pattern as well as some extra information on some of the transitions made during the execution of $\widetilde{\Pi}$. We also replace *hash collisions* with *hash mismatches* which are a notion of inconsistency of hash functions that includes hash collisions as a special case. The advantage of these notions is that now hash mismatches depend *entirely* on the behavior pattern and the randomness string.

We focus on a certain subset of behavior patterns we name *typical behavior patterns*; those are a subset of the behavior patterns that can occur when the adversary is limited to ϵ fraction of errors. We then show that there are at most $2^{O(H(\epsilon)n)} = 2^{O(\log(1/\epsilon)\epsilon n)}$ different typical behavior patterns, and that for each typical behavior pattern, at least a $1 - 2^{-\Omega(c'\log(1/\epsilon)\epsilon n)}$ fraction of the randomness strings lead to at most ϵn hash mismatches. This implies in turn that for a large enough constant c' there must exist a *single* good randomness string that leads to at most ϵn hash mismatches (and thus, at most ϵn hash collisions) for *all* typical behavior patterns. So this good randomness string leads to a successful simulation whenever the adversary is limited to flipping at most a fraction ϵ of the bits. Details follow.

6.2.1 Behavior patterns and hash mismatches

We start by formally defining the notions of behavior patterns and hash mismatches and proving that hash mismatches depend only on the behavior pattern and the randomness string.

Definition 6.1. (Behavior pattern) *Let Γ be a (possibly partial) run of the coding scheme $\widetilde{\Pi}$ (determined by the randomness string, the simulated noiseless*

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

protocol π, the inputs (x, y) of the parties and the noise pattern). The behavior pattern $\mathcal{P}$ of Γ consists of the following information:

1. *The number of iterations in Γ.*
2. *The noise pattern in Γ (that is, the communication rounds in Γ in which the channel flipped a bit).*
3. *The iterations in Γ in which no block was added to T_A and the iterations in Γ in which no block was added to T_B.*
4. *For each of the iterations in Γ in which no block was added to T_A, a bit saying whether Alice made a transition on Line 25, a bit saying whether Alice returned to* MP1 *on Line 27 and a bit saying whether Alice returned to* MP2 *on Line 30. Similarly, for each of the iterations in Γ in which no block was added to T_B, a bit saying whether Bob made a transition on Line 25, a bit saying whether Bob returned to* MP1 *on Line 27 and a bit saying whether Bob returned to* MP2 *on Line 30.*

DEFINITION 6.2. (HASH MISMATCH) *Let $i \in [R_{\text{total}}]$ be some iteration, let S be the seed used at iteration i, and let k_A, $|T_A|$, $N'(T_A)$, MP1_A and MP2_A (respectively, k_B, $|T_B|$, $N'(T_B)$, MP1_B and MP2_B) be the values of the variables of Alice (respectively, Bob) at the beginning of iteration i. Let $e \in \{0,1\}^{|T_A|}$ be the vector that indicates the locations of the adversarial errors in the communication rounds in which the bits of T_A were transmitted. We say that a hash mismatch occurred at iteration i if at least one of the following occurred at iteration i.*

1. *$k_A \neq k_B$ but $\text{hash}_S(k_A) = \text{hash}_S(k_B)$.*
2. *$e \neq 0$ but $h_S(e) = 0$.*
3. *$|T_A| \neq |T_B|$ but $h_S(|T_A|) = h_S(|T_B|)$.*
4. *$N'(T_A) \neq N'(T_B)$ but $h_S(N'(T_A)) = h_S(N'(T_B))$.*
5. *There exists $b \in \{1,2\}$ such that $e[1, \text{MP}b_A] \neq 0$ but $h_S(e[1, \text{MP}b_A]) = 0$.*
6. *There exist $b, b' \in \{1,2\}$ such that $\text{MP}b_A \neq \text{MP}b'_B$ but $h_S(\text{MP}b_A) = h_S(\text{MP}b'_B)$.*
7. *There exist $b, b' \in \{1,2\}$ such that $N'(T_A[1, \text{MP}b_A]) \neq N'(T_B[1, \text{MP}b'_B])$ but $h_S(N'(T_A[1, \text{MP}b_A])) = h_S(N'(T_B[1, \text{MP}b'_B]))$.*

The following claim says that if some iteration does not suffer from a hash mismatch then it does not suffer from a hash collision either.

CLAIM 6.1. *If an iteration of $\widetilde{\Pi}$ does not suffer from a hash mismatch then it does not suffer from a hash collision.*

6.2.2 Existence of good randomness string In this section we show the existence of a good random string R^* that can be used to derandomize the coding scheme $\widetilde{\Pi}$. For this we shall use the notion of a typical behavior pattern defined as follows.

DEFINITION 6.3. (TYPICAL BEHAVIOR PATTERN) *We say that a behavior pattern $\mathcal{P}$ is* typical *if the number of bit flips in the noise pattern of $\mathcal{P}$ is at most $2\epsilon n$, the number of iterations in $\mathcal{P}$ in which no block was added to T_A is at most $100\epsilon n$, and the number of iterations in $\mathcal{P}$ in which no block was added T_B is at most $100\epsilon n$.*

Using a counting argument, we can bound the number of typical behavior patterns.

CLAIM 6.2. *There are at most $2^{900H(\epsilon)n}$ different typical behavior patterns.*

Next we show that for every behavior pattern most randomness strings lead to at most ϵn hash mismatches.

CLAIM 6.3. *Let $\mathcal{P}$ be a behavior pattern and let R be a random string sampled as in Line 5. Then with probability at least $1 - 2^{-\Omega(c' \log(1/\epsilon)\epsilon n)}$ the number of iterations suffering from hash mismatches determined by $\mathcal{P}$ and R is at most ϵn.*

Claims 6.2 and 6.3 above imply the existence of a single random string R^* that leads to at most ϵn hash mismatches for all typical behavior patterns.

COROLLARY 6.1. *For sufficiently large constant c', there is a string $R^* \in \{0,1\}^{R_{\text{total}} \cdot s}$ such that for every typical behavior pattern $\mathcal{P}$ the number of iterations suffering from hash mismatches determined by $\mathcal{P}$ and R^* is at most ϵn.*

The fact that at most ϵn hash mismatches (and therefore, hash collisions) have occurred, leads to a successful simulation of $\widetilde{\Pi}$. This follows from the next Lemma which can be proved along the lines of the proof of Theorem 7.1 in [15].

LEMMA 6.1. *Let $R \in \{0,1\}^{R_{\text{total}} \cdot s}$ be an arbitrary string (not necessarily coming from a δ-biased distribution), and let Γ be a run of $\widetilde{\Pi}$ that uses the string R as the random string sampled at Line 5, and simulates a noiseless protocol π on inputs (x, y) in the presence of up to ϵ fraction of adversarial errors. Suppose furthermore that at most ϵn iterations in Γ suffer from a hash collision. Then the output of Γ is $\pi(x, y)$ (that is, the simulation performed by Γ is successful).*

It then follows that when the coding scheme $\widetilde{\Pi}$ is run with the random string R^* guaranteed by the

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

above corollary then the number of iterations suffering from *hash collisions* is at most ϵn, and the simulation is successful.

CLAIM 6.4. *Let $R^* \in \{0,1\}^{R_{\text{total}} \cdot s}$ be a string such that for every typical behavior pattern $\mathcal{P}$ the number of iterations suffering from hash mismatches determined by $\mathcal{P}$ and R^* is at most ϵn. Let Γ be a run of $\widetilde{\Pi}$ that uses the string R^* as the random string sampled at Line 5 and has at most ϵ fraction of adversarial errors. Then at most ϵn iterations in Γ suffer from a hash collision, and the simulation is successful.*

The above leads to Lemma 3.3, up to the obtained rate of the scheme which can easily be seen to be $1 - O(\sqrt{H(\epsilon)})$. Finding R^* takes $2^{O(n)}$ operations by exhaustive search. See full version for detailed proofs.

References

[1] SHWETA AGRAWAL, RAN GELLES, AND AMIT SAHAI, *Adaptive protocols for interactive communication*. Manuscript, arXiv:1312.4182 (cs.DS), 2013.

[2] ZVIKA BRAKERSKI, YAEL TAUMAN KALAI, AND MONI NAOR, *Fast interactive coding against adversarial noise*, J. ACM, 61 (2014), pp. 35:1–35:30.

[3] MARK BRAVERMAN, *Towards deterministic tree code constructions*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, ACM, 2012, pp. 161–167.

[4] MARK BRAVERMAN AND KLIM EFREMENKO, *List and unique coding for interactive communication in the presence of adversarial noise*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, FOCS '14, 2014, pp. 236–245.

[5] MARK BRAVERMAN, RAN GELLES, JIEMING MAO, AND RAFAIL OSTROVSKY, *Coding for interactive communication correcting insertions and deletions*. Manuscript, arXiv:1508.00514 (cs.DS), 2015.

[6] M. BRAVERMAN AND A. RAO, *Toward coding for maximum errors in interactive communication*, Information Theory, IEEE Transactions on, 60 (2014), pp. 7248–7255.

[7] URIEL FEIGE AND JOE KILIAN, *Finding OR in a noisy broadcast network*, Information Processing Letters, 73 (2000), pp. 69–75.

[8] G. DAVID FORNEY, *Concatenated codes*, Tech. Report 440, Massachusetts Institute of Technology. Research Laboratory of Electronics, 1965.

[9] MATTHEW FRANKLIN, RAN GELLES, RAFAIL OSTROVSKY, AND LEONARD J. SCHULMAN, *Optimal coding for streaming authentication and interactive communication*, Information Theory, IEEE Transactions on, 61 (2015), pp. 133–145.

[10] RAN GELLES, ANKUR MOITRA, AND AMIT SAHAI, *Efficient coding for interactive communication*, Information Theory, IEEE Transactions on, 60 (2014), pp. 1899–1913.

[11] MOHSEN GHAFFARI AND BERNHARD HAEUPLER, *Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, FOCS '14, 2014, pp. 394–403.

[12] MOHSEN GHAFFARI, BERNHARD HAEUPLER, AND MADHU SUDAN, *Optimal error rates for interactive coding I: Adaptivity and other settings*, in Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14, ACM, 2014, pp. 794–803.

[13] EDGAR N. GILBERT, *A comparision of signalling alphabets*, Bell System Technical Journal, 31 (1952), pp. 504–522.

[14] BERNHARD HAEUPLER, *Interactive channel capacity revisited*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, FOCS '14, 2014, pp. 226–235.

[15] BERNHARD HAEUPLER, *Interactive channel capacity revisited*, 2014. Long version of [14], [online:] http://arxiv.org/abs/1408.1467.

[16] RICHARD W. HAMMING, *Error detecting and error correcting codes*, Bell System technical journal, 29 (1950), pp. 147–160.

[17] GILLAT KOL AND RAN RAZ, *Interactive channel capacity*, in STOC '13: Proceedings of the 45th annual ACM Symposium on Theory of Computing, ACM, 2013, pp. 715–724.

[18] IRVING S. REED AND GUSTAVE SOLOMON, *Polynomial codes over certain finite fields*, SIAM Journal of the Society for Industrial and Applied Mathematics, 8 (1960), pp. 300–304.

[19] LEONARD J. SCHULMAN, *Communication on noisy channels: a coding theorem for computation*, Foundations of Computer Science, Annual IEEE Symposium on, 1992, pp. 724–733.

[20] LEONARD J. SCHULMAN, *Deterministic coding for interactive communication*, in STOC '93: Proceedings of the twenty-fifth annual ACM Symposium on Theory of Computing, ACM, 1993, pp. 747–756.

[21] LEONARD J. SCHULMAN, *Coding for interactive communication*, IEEE Transactions on Information Theory, 42 (1996), pp. 1745–1756.

[22] LEONARD J. SCHULMAN, *A postcript to "coding for interactive communication"*. [Online:] http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt, 2003. based on joint work with Will Evans and Michael Klugerman.

[23] CLAUDE E. SHANNON, *A mathematical theory of communication*, ACM SIGMOBILE Mobile Computing and Communications Review, 5 (2001), pp. 3–55.

[24] R. R. VARSHAMOV, *Estimate of the number of signals in error correcting codes*, Doklady Akadamii Nauk, (1957), pp. 739–741.

[25] ANDREW C.-C. YAO, *Some complexity questions related to distributive computing(preliminary report)*, in Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, STOC '79, ACM, 1979, pp. 209–213.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.