Explicit Capacity Approaching Coding for Interactive Communication

Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson

***Abstract*— We show an *explicit* (that is, efficient and deterministic) capacity approaching interactive coding scheme that simulates any interactive protocol under random errors with nearly optimal communication rate. Specifically, over the binary symmetric channel with crossover probability ϵ, our coding scheme achieves a communication rate of $1-O(\sqrt{H(\epsilon)})$, together with negligible $\exp(-\Omega(\epsilon^4\ n/\log n))$ failure probability (over the randomness of the channel). A rate of $1-\tilde{\Theta}(\sqrt{H(\epsilon)})$ is likely asymptotically optimal as a result of Kol and Raz (2013) suggests. Prior to this paper, such a communication rate was achievable only using *randomized* coding schemes [Kol and Raz (2013); Hauepler (2014)].**

***Index Terms*— Interactive communication, coding protocols, tree codes, deterministic interactive coding.**

I. Introduction

A. 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 Hamming's [26] and Shannon's [34] breakthrough papers, 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 [36] 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 [30]–[32] 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 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 explicit 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* coding 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., [1], [2], [4]–[7], [9]–[11], [14], [16]–[19], [21], [23]–[25], [27] and the survey [15] by Gelles. To informally cite but one central recent result, Kol and Raz [27] (see also [21]) proved that for the binary symmetric channel BSC_ϵ, in which every communicated bit is flipped independently with probability ϵ, the communication rate for certain protocols 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 $1-H(\epsilon)$. Kol and Raz [27] and Haeupler [21] also gave efficient *randomized* coding schemes for the BSC_ϵ with rate $1-\Theta(\sqrt{H(\epsilon)})$ and $1-\Theta(\sqrt{\epsilon})$ respectively.

Next we describe the basic protocol structure and coding problem more precisely. We use the standard model of alternating, deterministic protocols. Assume that Alice and

Manuscript received April 26, 2017; revised January 18, 2018; accepted April 2, 2018. Date of publication April 24, 2018; date of current version September 13, 2018. R. Gelles was supported in part by NSF under Grant CCF-1149888 and in part by the Israel Science Foundation under Grant 1078/17. B. Haeupler was supported in part by the NSF under Grant CCF-1527110 and Grant CCF-1618280, and in part by the NSF CAREER award under Grant CCF-1750808. G. Kol was supported in part by the Fund for Math and in part by the Weizmann Institute of Science National Post-Doctoral Award Program for Advancing Women in Science. N. Ron-Zewi was supported in part by NSF under Grant CCF-1412958 and Grant CCF-1445755, in part by a Rothschild fellowship, and in part by an Alon fellowship. A. Wigderson was supported by the NSF under Grant CCF-1412958. This paper was presented at the 2016 Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms.

R. Gelles is with the Faculty of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel (e-mail: ran.gelles@biu.ac.il)

B. Haeupler is with the Department of computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA (e-mail: haeupler@cs.cmu.edu).

G. Kol is with the Department of Computer Science, Princeton University, NJ 08540, USA (e-mail: gkol@cs.princeton.edu).

N. Ron-Zewi is with the Department of Computer Science, University of Haifa, Haifa 3498838, Israel (e-mail: noga@cs.haifa.ac.il).

A. Wigderson is with the School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540, USA (e-mail: avi@ias.edu).

Communicated by P. Gopalan, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2018.2829764

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 deterministic communication protocol π for this task, in which the parties alternate sending bits: 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 essentially the standard communication complexity setting.

Now assume that the communication channel is noisy. Our main result is concerned with the probabilistic noise model but throughout we shall consider also the adversarial noise model. 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 approach the capacity of the channel. Another important goal is explicitness. Ideally the coding scheme should be deterministic,[1] and the computational complexity of the parties using the scheme Π (given π and their inputs) should be at most polynomial in $|\pi|$. In this work we focus on the probabilistic noise model BSC_ϵ which is the original model studied in [30]. We leave the question of generalizing our results to the adversarial setting as a significant challenge for future research.

B. Problem Statement and Main result

1) Problem Statement: Let X, Y be finite sets, and let π be a deterministic two-party interactive protocol between Alice and Bob that assumes reliable channels. For any $x \in X$ given to Alice and any $y \in Y$ given to Bob, the protocol π communicates $n = |\pi|$ bits, alternating between the parties, after which both parties output $\pi(x, y)$.

Next, the communication channel is replaced with BSC_ϵ for some fixed constant $\epsilon < 1/2$. The task we wish to solve is the following: given any protocol π as above, devise a *deterministic* protocol Π that *simulates* π, that is, for any $x \in X$ given to Alice and any $y \in Y$ given to Bob, the "coding scheme" Π communicates $N = |\Pi|$ bits, alternating between the parties, after which both parties output $\pi(x, y)$; The coding scheme Π should have (i) a good communication rate n/N, ideally $1 - \tilde{O}(\sqrt{H(\epsilon)})$, (ii) a deterministic time complexity which is polynomial in n, and (iii) a negligible failure probability, ideally exponential in n but at most $n^{-\omega(1)}$ over the randomness of the channel.

2) Prior State of the Art: As mentioned above, this line of work was initiated in [30], which gave an efficient *randomized* coding scheme of (small) constant rate over BSC_ϵ assuming the parties share a common random string; this assumption was recently eliminated in [16]. The follow-up work [31] gave also a *deterministic* scheme of (small) constant rate (bounded away from 1) (small) for both the probabilistic and adversarial error models. However, this latter scheme was based on tree codes for which no efficient construction is currently known, and therefore it is non-efficient.

In recent years there have been quite a few advancements on obtaining efficient *randomized* coding schemes with optimal parameters. Specifically [4] obtained the first efficient randomized scheme of (small) constant rate in the adversarial error model; Later, Ghaffari and Hauepler [18] achieved in the adversarial setting an efficient randomized scheme with optimal resilience and (small) constant rate. For the case where the noise level is low, $\epsilon \to 0$, Kol and Raz [27] showed that the capacity of BSC_ϵ in the interactive setting is $1 - \widetilde{\Theta}(\sqrt{H(\epsilon)})$, and gave an efficient randomized coding scheme with rate $1 - \Theta(\sqrt{H(\epsilon)})$ over BSC_ϵ channels; Haeupler [21] obtained an improved efficient randomized scheme that achieves a rate of $1 - \Theta(\sqrt{\epsilon})$ over BSC_ϵ channels. The scheme of [21] also works for adversarial channels corrupting any ϵ fraction of transmission with a rate of $1 - \Theta(\sqrt{\epsilon \log\log 1/\epsilon})$. Both rates are conjectured to be optimal [22]. As to efficient *deterministic* coding schemes, [5] gave an efficient deterministic scheme of (small) constant rate over the BSC_ϵ with failure probability $\exp(-\log^{\Omega(1)} n)$ by providing a sub-exponential time tree code construction and applying any tree-code based deterministic coding scheme separately on blocks of polylogarithmic length. Subsequent to this work Cohen *et al.* [11] gave an explicit deterministic binary tree code construction over a polylogarithmic size alphabet. Since this tree code does not have an efficient decoding algorithm it does currently not yet lead to an improved interactive coding scheme for the BSC_ϵ.

3) Main Result: Our main result provides the first interactive coding scheme for BSC_ϵ channel which is both efficient, deterministic and approaching the capacity.

Theorem 1 (Main): For every sufficiently small constant $\epsilon > 0$ and every sufficiently large n (as a function of ϵ), there exists an efficient deterministic interactive coding scheme Π that simulates any noiseless protocol π of length n over

[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., [12]. In particular, Π must be correct also if there is no noise at all.

BSC_ϵ *with rate* $1 - O\left(\sqrt{H(\epsilon)}\right)$ *and failure probability* $\exp(-\Omega(\epsilon^4\, n/\log n))$.

As mentioned above, a rate of $1-\tilde{\Theta}(\sqrt{H(\epsilon)})$ is conjectured to be optimal due to the results of [27]. Obtaining an efficient deterministic coding scheme with similar rate (or even with some (small) constant rate) for an *adversarial channel* remains a challenging problem.

C. Overview of techniques

Our coding scheme exploits the idea of *code concatenation* [13], which is a very common (simple yet powerful) technique in the one-way coding theory literature, and our main conceptual contribution is an adaptation of this technique to the interactive setting.[2]

Concatenation usually consists of two separate coding layers: an *inner code* which is defined over binary alphabet and may be inefficient, and an *outer code* which must be efficient and is defined over large alphabet. In the standard concatenation approach one first splits the binary message into short blocks, say of length $O(\log n)$, views each block as a single symbol in a large alphabet, and encodes the message via the outer code where each block is considered as a single input symbol for the outer code. Then one switches view again and thinks of each large output symbol of the outer code as a binary string, and encodes each such string separately via the inner code. Such concatenation results in an efficient binary code, and by choosing the right parameters one can also guarantee that the final code has high rate and low failure probability.

More concretely, in the typical concatenation setting one chooses the outer code to be a code of nearly optimal rate $\rho_{\text{out}} \approx 1$ that is resilient to some δ_{out} fraction of adversarial errors. The inner code on the other hand is chosen to be a code of some rate ρ_{in} that has optimal exponentially small failure probability over BSC_ϵ. It can be verified that the rate of the final code is the product $\rho_{\text{out}} \cdot \rho_{\text{in}}$ of the rates of the outer and inner codes, and since we chose the rate ρ_{out} of the outer code to be close to 1 then the rate of the final code would be roughly the rate of the inner code ρ_{in}. Furthermore, the final code fails only if more than δ_{out} fraction of the inner codes fail, which for sufficiently large δ_{out} happens with probability at most $(2^{-\Omega(\log n)})^{\delta_{\text{out}}(n/\log n)} = 2^{-\Omega(\delta_{\text{out}} \cdot n)}$ over BSC_ϵ. To summarize, in the above setting of parameters the final code inherits on one hand the running time and the resilience of the outer code, and on the other hand the alphabet size and the rate of the inner code.

Towards an interactive version of concatenation, we take a careful examination of the properties that the outer and inner codes should satisfy in order to enable interactive concatenation. As it turns out, assuming that both the outer and inner codes are interactive coding schemes, the only other property that is required for interactive concatenation is that the outer code could be encoded online when viewing both its input and output symbols as *binary* strings. This property is crucial in the interactive setting since the original protocol is an interactive protocol over binary alphabet and therefore one cannot encode a large chunk of it a-priori before it was communicated. One way to ensure the online encoding property is to insist that the outer code would be *systematic*, which means that for every output symbol y_i it holds that $y_i = (x_i, r_i)$ where x_i is the i-th input symbol (the "systematic" part) and r_i is the "redundant" part that may depend on all previous input symbols. Indeed, if this is the case, then the parties can first communicate x_i via the original protocol in an online fashion, and then communicate r_i which depends only on previous input symbols. As linear codes can always be made systematic it in fact suffices that the outer code would be *linear*. We believe that this observation may be useful in the future for the design of other interactive coding schemes.

For simplicity, in our concatenation lemma (Lemma 4) we show how to implement the above program for the special case in which the outer code is a *tree code* based interactive coding scheme[3] (and the inner code is an arbitrary ineractive coding scheme). To obtain our final construction, as the outer code we use an efficiently encodable and decodable *linear* tree code of relative distance $\Omega(1/\log n)$ and rate $\approx$1, defined over an $O(\log n)$-bit alphabet. As the inner code we use an exponential time deterministic interactive coding scheme over binary alphabet that can correct ϵ fraction of adversarial errors (and so has exponentially small failure probability over BSC_ϵ) with rate $1 - O(\sqrt{H(\epsilon)})$. The resulting coding scheme is then an efficient interactive coding scheme of rate $1 - O(\sqrt{H(\epsilon)})$ and failure probability $\exp(-\Omega(\epsilon^4\, n/\log n))$ over BSC_ϵ.

In what follows we describe the outer and inner codes we use in more detail and the techniques used to obtain them. We start with the inner code construction which is more technically involved.

1) Inner Code: The inner code construction is based on an efficient *randomized* coding scheme by Haeupler [21]. That scheme achieves a rate of $1 - O(\sqrt{\epsilon})$ over a BSC_ϵ (together with exponentially small failure probability), however it is randomized. We obtain our inner code by derandomizing this scheme, and the main technical challenge 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 [4], however it is more involved due to the need to maintain high rate in derandomization, and the intricacy of the coding scheme of [21] and its analysis. In more detail, the main use of randomness in both schemes [21], [4] 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, the parties choose a random hash function and communicate short hashes of their own states of length $\ell \ll n$. Note that due to this shrinkage in length, it may happen that although the parties are

[2]The unpublished memo [33] and the work [5] constructed tree codes using layers of codes of different lengths combined together via "pointwise concatenation" (see "Outer Code" paragraph below for an illustration of this approach); However, this seems rather different from our concatenation approach whose purpose is to turn an optimal non-efficient inner code and an efficient non-optimal outer code into an efficient and optimal coding scheme.

[3]We note however that currently essentially all known deterministics interactive coding schemes are based on tree codes or their variants.

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*.

In [4], Brakerski *et al.* first 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. They then show that for each such sequence, at least a $1-2^{-\Omega(\ell\epsilon n)}$ 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. Choosing $\ell=\Omega(1/\epsilon)$, 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 to communicate hashes of length $\Omega(1/\epsilon)$. To this end, we observe that when the adversary is limited to corrupting at most a fraction ϵ of the transmissions, then 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 [21] 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 actually creates further complications in our proof which we ignore for now, see Section V-B for more details).

We then show that for each typical noise pattern, at least a $1-2^{-\Omega(\ell\epsilon n)}$ fraction of the randomness strings lead to at most ϵn hash collisions. Consequently, choosing $\ell=\Omega(\log(1/\epsilon))$, a union bound argument on all the possible noise patterns proves the existence of a single random string that works for any typical noise pattern. This results in a slightly reduced rate of $1-O(\sqrt{H(\epsilon)})$ compared to [21] in which ℓ was chosen to be some constant independent of ϵ.

2) Outer Code: The starting point for the outer code construction is a tree code construction outlined in an unpublished memo by Schulman [33]. That construction uses the idea of encoding substrings 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. However, since the codes used in [33] had rate bounded away from 1 the resulting tree code had rate $O(1/\log n)$; on the other hand, the obtained relative distance was constant $\Omega(1)$.

For our application, we require that the tree code has high rate very close to 1, but are willing to tolerate low (even slightly sub-constant) distance. To deal with this, we first replace the codes used in [33] with codes of rate $1-O(1/\log n)=1-o(1)$ and relative distance roughly $1/\log n=o(1)$. Furthermore, to guarantee high rate we also use *systematic* codes and spread only the *redundant* part across the next 2^t levels. The resulting tree then has rate ≈ 1, relative distance roughly $1/\log n$, and $O(\log n)$-bit alphabet. Moreover, if the codes used are linear then so is the tree code.

To turn the tree code described above into an interactive coding scheme we use in our concatenation lemma (Lemma 4) a scheme similar to the tree code based scheme of [31]. However, the original analysis of [31] only achieved a constant rate bounded away from one,[4] regardless of the rate of the tree code, and we provide a slightly tighter analysis of this scheme that preserves the rate of the tree code. We also observe that the coding scheme preserves the linearity of the tree code, and consequently this scheme could be used as the outer code in our concatenation scheme.

D. Organization of the Paper

We begin (Section II) by recalling several building blocks and setting up notations we will use throughout. Our main concatenation lemma is provided in Section III, along with a formal statement of the inner and outer codes we use. These prove our main Theorem 1. The detailed proof of the concatenation lemma appears in Section IV. In Sections V and VI we present our inner and outer code constructions, respectively.

II. 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 $m\le n$ let $[m,n]:=\{m,m+1,\ldots,n\}$. For a vector $x\in\Sigma^n$ and integers $1\le i\le j\le n$ 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 $\mathrm{dist}(x,y):=|\{i\in[n]:x_i\neq y_i\}|/n$.

A. Error-Correcting Codes

A code is a mapping $C:\Sigma_{\text{in}}^k\to\Sigma_{\text{out}}^n$. 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 $\frac{k\log|\Sigma_{\text{in}}|}{n\log|\Sigma_{\text{out}}|}$. We say that C has *relative distance* at least δ if for every pair of distinct vectors $x,y\in\Sigma_{\text{in}}^k$ it holds that $\mathrm{dist}(C(x),C(y))\ge\delta$.

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 $\Sigma_{\text{in}}=\Sigma_{\text{out}}=\mathbb{F}$ and C is $\mathbb{F}$-linear then we simply say that C is *linear*. Finally, if $k=n$ then we say that a code $C:\Sigma_{\text{in}}^n\to\Sigma_{\text{out}}^n$ is *systematic* if $\Sigma_{\text{in}}=\Gamma^s$ and $\Sigma_{\text{out}}=\Gamma^{s+r}$ 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 $x\in\Sigma_{\text{in}}^n$ 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.

[4]Indeed, the tree code based scheme in [33] is run for $5|\pi|$ rounds, so rate is at most $1/5$.

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.

Fact 2 (Reed-Solomon Codes [29]): 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 fact 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 [20], [35].

Fact 3 (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 $\mathrm{poly}(n)$.

B. Tree Codes

A tree code [32] 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* [14] (see also [6] and [8]). The *suffix distance* between a pair of strings $x, y \in \Sigma^n$ 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 *(efficiently) decoded from α fraction of errors* if there exists a polynomial time decoding algorithm $\mathtt{dec}_\Lambda$ which takes as input a vector $w \in \Sigma_{\mathrm{out}}^l$ for any length $l \in [n]$ and outputs a vector $y \in \Sigma_{\mathrm{in}}^l$ of the same length such that if w is α-close to some codeword $\Lambda(x)$ in suffix distance then the decoding algorithm recovers x from w, i.e., $\forall l \in [n], w \in \Sigma_{\mathrm{out}}^l, x \in \Sigma_{\mathrm{in}}^l$: If $\mathrm{dist}_{\mathrm{sfx}}\big(w, \Lambda(x)\big) \leq \alpha$, then $\mathtt{dec}_\Lambda(w) = x$.

III. Explicit Capacity Approaching Coding Scheme

In this section we prove our main Theorem 1. This theorem is 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 IV). Then, in Lemmas 5 and 6 below, we state the existence of inner and outer codes with good parameters, whose concatenation proves our main Theorem 1. Recall the definition of 'simulation' given in Section I-B.

Lemma 4 (Concatenation): Suppose that the following hold:

1) *(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 $\mathcal{T}_\Pi$.*
2) *(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 $\mathcal{T}_\Lambda$.*

Then for every $\gamma > 0$ there exists a deterministic interactive coding scheme Π' that simulates any noiseless protocol π' over $\mathsf{BSC}_{\delta_\Pi/2}$ 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 failure probability

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

Furthermore, the coding scheme Π' has running time

$$O(n_\Lambda \cdot (\mathcal{T}_\Lambda + \mathcal{T}_\Pi)).$$

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

Lemma 5 (Inner Code): For every sufficiently small constant $\epsilon > 0$ there exists a deterministic interactive coding scheme Π that simulates any noiseless protocol π of length n with rate $1 - O\left(\sqrt{H(\epsilon)}\right)$ in the presence of up to a fraction ϵ 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 V.

Lemma 6 (Outer Code): There exists an absolute constant $\delta_0 > 0$ such that the following holds for every sufficiently small constant $\epsilon > 0$ and every sufficiently large n such that $\epsilon > \frac{\Omega(\log n)}{n}$. 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 a fraction* $\frac{\delta_0\cdot\epsilon}{2\log n}$ *of errors in time* $\mathrm{poly}(n)$.

We prove the above lemma in Section VI. We can now prove our main Theorem 1 based on the above Lemmas 4, 5 and 6.

Proof of Theorem 1: Given any constant $\epsilon > 0$ and a sufficiently large length n, let Λ be the tree code guaranteed by Lemma 6 for ϵ and integer n_Λ such that $n = n_\Lambda((\log n_\Lambda)/\epsilon - 2)(1-\epsilon)$ (so $n_\Lambda = \Omega(\epsilon n/\log n)$). In particular, Λ satisfies the outer code requirement of Lemma 4 with $s = (\log n)/\epsilon$, $r = \log n$, $\rho_\Lambda = \frac{1}{1+\epsilon}$, $\delta_\Lambda = \frac{\delta_0\cdot\epsilon}{2\log n}$, and $\mathcal{T}_\Lambda = n^{O(1)}$. Now let Π be the coding scheme guaranteed by Lemma 5 for a $\delta_\Pi = 2\epsilon$ fraction of errors simulating protocols of length $n' = s + (2r+1) = \Theta((\log n)/\epsilon)$ with rate $\rho_\Pi = 1 - O(\sqrt{H(\epsilon)})$ and construction time $\mathcal{T}_\Lambda = 2^{n'} = n^{O(\epsilon)}$. This Π satisfies the inner code requirement of Lemma 4. For $\gamma = \epsilon$, Lemma 4 now proves Theorem 1. In particular, the coding scheme Π' simulates any noiseless protocol of length n over $\mathsf{BSC}_{\delta_\Pi/2} = \mathsf{BSC}_\epsilon$ with failure probability

$$\exp\left[-\Omega\left(n_\Lambda\left(\tfrac{\delta_\Lambda}{36}\cdot\tfrac{s+2(r+1)}{\rho_\Pi}\cdot\tfrac{\delta_\Pi^2}{4}\cdot\gamma - H\left(\tfrac{\delta_\Lambda}{36}\cdot\gamma\right)\right)\right)\right] = \exp\left[-\Omega\left(\epsilon^4\, n/\log n\right)\right],$$

a running time of

$$O(n_\Lambda\cdot\ (\mathcal{T}_\Lambda + \mathcal{T}_\Pi)) = O(O(n)\cdot\ (n^{O(1)} + n^{O(1/\epsilon)})) = n^{O(1/\epsilon)}$$

and rate

$$\frac{\rho_\Lambda}{2-\rho_\Lambda+4/(s-2)}\cdot\rho_\Pi\cdot\ (1-\gamma) = \frac{\frac{1}{1+\epsilon}}{2-\frac{1}{1+\epsilon}+4/((\log n)/\epsilon-2)}\cdot\left(1-O(\sqrt{H(\epsilon)})\right)\cdot\ (1-\epsilon) = 1-O(\sqrt{H(\epsilon)}).$$

■

IV. The Concatenation Lemma: Proof of Lemma 4

The coding scheme Π' is similar to the tree code based scheme of [32], where we replace each input symbol to the tree code with an inner code simulation (treated as a single symbol over large alphabet). We stress again that this is possible since the tree code is systematic, and so one can view each input symbol obtained by the inner code simulation as the prefix of the corresponding output symbol.

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* $p_{i-1}^A, p_{i-1}^B \in \{0,1\}^2$, and a pair of *hash strings* $h_{i-1}^A, h_{i-1}^B \in \{0,1\}^r$.

Alice (respectively, Bob) then performs, one of three actions according to the output of the simulation via the inner coding scheme Π: she either appends her noisy version X_i^A 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. These actions correspond to the case where a seemingly correct simulation of X_i has occurred, a seemingly incorrect simulation has occurred, or it seems that the prefix T^A is incorrect, respectively. She then records her action in the i-th position string p_i^A (since there are only three possible actions those could be recorded using 2 bits).

Lastly, Alice views the string $(\sigma_{\text{in}})_i^A := (p_i^A, X_i^A) \in \{0,1\}^s$ as the systematic part of the i-th output symbol of the tree code Λ and lets $(\sigma_{\text{out}})_i^A$ be the corresponding i-th output symbol of the tree code. The i-th hash string $h_i^A \in \{0,1\}^r$ is set to be the redundant part of $(\sigma_{\text{out}})_i^A$. As described above, both the strings p_i^A and h_i^A will be communicated by Alice in iteration $i+1$. Note that for every i, the string $((\sigma_{\text{in}})_1^A, \ldots, (\sigma_{\text{in}})_i^A)$ 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.

A. 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_{i-1}^A denotes the first 2 bits of $(\sigma_{\text{out}})_{i-1}^A$ and let h_{i-1}^A denote the last r bits of $(\sigma_{\text{out}})_{i-1}^A$ (for $i=1$ let $(\sigma_{\text{out}})_0^A := 0^{s+r}$).
2) Simulate the protocol $\pi^A\left(|T^A|, (T^A, 0^{n_\Lambda(s-2)-|T^A|}), p_{i-1}^A, h_{i-1}^A\right)$ below using the inner coding scheme Π. Let the sequence $(p_{i-1}^A, \hat{p}_{i-1}^B, h_{i-1}^A, \hat{h}_{i-1}^B, X_i^A)$ denote the output of the simulation where $p_{i-1}^A, \hat{p}_{i-1}^B \in \{0,1\}^2$, $h_{i-1}^A, \hat{h}_{i-1}^B \in \{0,1\}^r$ and $X_i^A \in \{0,1\}^{s-2}$.
3) Let $(\widehat{\sigma}_{\text{out}})_{i-1}^B := \left(\hat{p}_{i-1}^B, X_{i-1}^A, \hat{h}_{i-1}^B\right)$. Decode the sequence $((\widehat{\sigma}_{\text{out}})_1^B, \ldots, (\widehat{\sigma}_{\text{out}})_{i-1}^B)$ using the decoding algorithm for Λ. Let $((\widehat{\sigma}_{\text{in}})_1^B, \ldots, (\widehat{\sigma}_{\text{in}})_{i-1}^B)$ 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_i^A to T^A and set $p_i^A := 01$.
5) Otherwise, if $T^A \neq \hat{T}^B$ and $|T^A| < |\hat{T}^B|$ set $p_i^A := 00$.
6) Otherwise, if $T^A \neq \hat{T}^B$ and $|T^A| \geq |\hat{T}^B|$ erase the last $s-2$ bits from T^A and set $p_i^A := 10$.
7) Let $(\sigma_{\text{in}})_i^A := (p_i^A, X_i^A)$ and let $(\sigma_{\text{out}})_i^A$ be the i-th symbol of $\Lambda((\sigma_{\text{in}})_1^A, \ldots, (\sigma_{\text{in}})_i^A)$. Note that since Λ is systematic it holds that $(\sigma_{\text{in}})_i^A$ is a prefix of $(\sigma_{\text{out}})_i^A$.

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.

B. Analysis

1) Rate and Running Time: The coding scheme Π' runs for n_Λ iterations and at each iteration the number of bits communicated is $((s-2)+2(r+2))/\rho_\Pi$. Recall that $\rho_\lambda = \frac{s}{s+r}$. Consequently, the rate of the coding scheme Π' is

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

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 $O(n_\Lambda \cdot (\mathcal{T}_\Lambda + \mathcal{T}_\Pi))$.

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 if at the end of the simulation it holds that $\Phi \geq n_\Lambda \cdot (1-\gamma)$, then the simulation must be successful. The reason is that in this case $t^+ \geq n_\Lambda \cdot (1-\gamma)$ and so a prefix of length at least $n_\Lambda \cdot (s-2) \cdot (1-\gamma)$ is correct in both T^A and T^B, which means the entire transcript π' was correctly simulated.

To bound the potential we shall use the notion of a *good iteration*.

Definition 7: 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 $T^B = \hat{T}^B$.*

Claim 8: The potential Φ decreases by at most 3 *after any iteration. Furthermore, after any* good iteration *the potential increases by at least* 1.

Proof: At any single iteration, a party either leaves its transcript unchanged, erases the last block of its transcript, or adds a new block to its transcript. Therefore t^+ can change by at most 1 and t^- can change by at most 2 at each iteration, and so the total potential change at each iteration is at most 3.

Next observe that if iteration i is good, then both parties know the transcript of the other side at the beginning of iteration; they also learn the correct value of the block X_i. Therefore, if $T^A = T^B$ at the beginning of iteration i, then both parties add X_i to their transcript, t^+ increases by 1 and t^- remains zero. Otherwise, if $T^A \neq T^B$ and $|T^A| = |T^B|$, then both parties erase the last block of their transcript, thus t^+ does not change and t^- decreases by 2. Finally, if $|T^A| \neq |T^B|$, then the party with the longer transcript erases the last block of its transcript and so t^+ does not change while t^- decreases by 1. We conclude that the total potential change at a good iteration is at least 1. ∎

Claim 8 above 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 $n_\Lambda \gamma/4$.

For our next claim we use a simple lemma whose statement and proof we take from [32].

Lemma 9 (Lemma 7 in [6]): In any finite set of intervals on the real line whose union J is of total length s there is a subset of disjoint intervals whose union is of total length at least $s/2$.

Proof: We show that J can be written as the union of two sequences of disjoint intervals. The question reduces to the case in which the intervals of the family are closed and their union J is an interval. In the first step put into the first sequence that interval which reaches the left endpoint of J, and which extends furthest to the right. In each successive step, select the interval which intersects the union of those selected so far, and which extends furthest to the right; adjoin the new interval to one of the sequences in alternation. ∎

Next we show that we can bound the number of bad iterations by bounding the number of iterations in which the first condition in Definition 7 does not hold.

Claim 10: If the first condition of Definition 7 does not hold in at most m iterations, then the number of bad iterations is at most $9m/\delta_\Lambda$.

Proof: By symmetry, it suffices to show that in addition to the m iteration where the first condition does not hold, 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. In particular, if the simulation fails at Step 2 in the segment $[t(i), i+1]$ at most m times, then $|[t(i), i+1]| < 2m/\delta_\Lambda$. However, we must take care of overlapping segments $[t(i), i+1]$.

Let

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

and define $\bigcup \mathcal{I} = \bigcup_{I \in \mathcal{I}} I$. Since for each iteration $i+1$ in which $T^B \neq \hat{T}^B$ it holds that $i+1 \in \bigcup \mathcal{I}$, it suffices to show that $\left|\bigcup \mathcal{I}\right| \leq 4m/\delta_\Lambda$. Lemma 9 shows that there exists a subset

$\mathcal{I}' \subseteq \mathcal{I}$ of disjoint intervals such that $\left|\bigcup \mathcal{I}'\right| \geq \left|\bigcup \mathcal{I}\right|/2$. The proof is completed by noting that if the simulation at Step 2 failed in at most m iterations, then it must be that $\left|\bigcup \mathcal{I}'\right| \leq 2m/\delta_\Lambda$, and so $\left|\bigcup \mathcal{I}\right| \leq 4m/\delta_\Lambda$. ∎

Using the above Claim 10, 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

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

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

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

V. The Inner Code: Proof of Lemma 5

The inner code is obtained via a derandomization of a randomized interactive coding scheme due to Haeupler [21, Algorithm 3]. We show how to devise a *deterministic* variant of the coding scheme of [21], 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.

Concretely, we prove Lemma 5 in two steps. In the first step we slightly modify the original scheme of [21], specifically, by carefully modifying the way the hash comparisons are performed, and slightly increasing the output length of the hash functions, as outlined in the introduction. In the second step we derandomize this coding scheme. The two steps are given in Sections V-A and V-B below, respectively. The proof and detailed analysis below builds modularly on the analysis of [21]; we only change the hashing part of the protocols and reprove in detail that our hashing scheme satisfies the exact properties needed for the correctness of coding scheme. In the following all the references (lemma numbers, line numbers, variable names, etc.) correspond to the full version [22] of [21].

A. The Modified Scheme $\widetilde{\Pi}$

In this section we slightly modify the hashing scheme in the randomized coding scheme given by [22, Algorithm 3] to obtain a randomized coding scheme $\widetilde{\Pi}$ that is more suitable for derandomization, and state some properties of the coding scheme $\widetilde{\Pi}$ that we shall use for the derandomization step. We start by describing $\widetilde{\Pi}$ and give its full pseudo-code in Algorithm 1, which keeps the line numbering the same as in [22, Algorithm 3].

Algorithm 1 The Coding Scheme $\widetilde{\Pi}$

1: $\Pi \leftarrow$ n-round protocol to be simulated + final confirmation steps
2: $\widetilde{\text{hash}}, \widetilde{\text{hash}} \leftarrow$ inner product hash family with $o = \Theta(\log 1/\epsilon)$ and $s = \Theta(n \log n)$
3: Initialize Parameters: $r_c \leftarrow \Theta(\log(1/\epsilon))$; $r \leftarrow \lceil\sqrt{\frac{r_c}{\epsilon}}\rceil$; $R_{total} \leftarrow \lceil n/r + 65\ n\epsilon\rceil$; $\mathtt{T} \leftarrow \emptyset$; $N(\cdot) \leftarrow \emptyset$
4: Reset Status: $\mathtt{k}, \mathtt{E}, \mathtt{v1}, \mathtt{v2} \leftarrow 0$
5: $R \leftarrow$ Random string of length $R_{total} \cdot s$ (can be constructed by exchanging $\Theta\left(\sqrt{\epsilon \log(1/\epsilon)}\right)n$ random bits and expanding them to a δ-bias string of the needed length using [3], [28], with bias $\delta = 2^{-\Theta(\frac{n}{r}o)}$)
6: **for** R_{total} iterations **do**
▷ **Verification Phase**
7: $\mathtt{k} \leftarrow \mathtt{k}+1$; $\tilde{\mathtt{k}} \leftarrow 2^{\lfloor \log_2 \mathtt{k} \rfloor}$; $\mathtt{MP1} \leftarrow \tilde{\mathtt{k}}r\lfloor\frac{|\mathtt{T}|}{\tilde{\mathtt{k}}r}\rfloor$; $\mathtt{MP2} \leftarrow \mathtt{MP1} - \tilde{\mathtt{k}}r$
8: $\mathtt{S} \leftarrow s$ new preshared random bits from R
9: Send $(\widetilde{\text{hash}}_\mathtt{S}(\mathtt{k}), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}[1,\mathtt{MP1}]), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}[1,\mathtt{MP2}]))$
10: Receive $(H'_\mathtt{k}, H'_\mathtt{T}, H'_{\mathtt{MP1}}, H'_{\mathtt{MP2}})$;
11: $(H_\mathtt{k}, H_\mathtt{T}, H_{\mathtt{MP1}}, H_{\mathtt{MP2}}) \leftarrow (\widetilde{\text{hash}}_\mathtt{S}(\mathtt{k}), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}[1,\mathtt{MP1}]), \widetilde{\text{hash}}_\mathtt{S}(\mathtt{T}[1,\mathtt{MP2}]))$
12: **if** $H_\mathtt{k} \neq H'_\mathtt{k}$ **then**
13: $\mathtt{E} \leftarrow \mathtt{E}+1$
14: **else**
15: **if** $H_{\mathtt{MP1}} \in \{H'_{\mathtt{MP1}}, H'_{\mathtt{MP2}}\}$ **then**
16: $\mathtt{v1} \leftarrow \mathtt{v1}+1$
17: **else if** $H_{\mathtt{MP2}} \in \{H'_{\mathtt{MP1}}, H'_{\mathtt{MP2}}\}$ **then**
18: $\mathtt{v2} \leftarrow \mathtt{v2}+1$
19: **if** $\mathtt{k}=1$ and $H_\mathtt{T} = H'_\mathtt{T}$ and $\mathtt{E}=0$ **then**
▷ **Computation Phase**
20: continue computation and transcript $\mathtt{T}$ for r steps (update $N(i)$ to indicate the round number in which the i-th bit of T was set.)
21: Reset Status: $\mathtt{k}, \mathtt{E}, \mathtt{v1}, \mathtt{v2} \leftarrow 0$
22: **else**
23: do r dummy communications
24: **if** $2\mathtt{E} \geq \mathtt{k}$ **then** ▷ **Transition Phase**
25: Reset Status: $\mathtt{k}, \mathtt{E}, \mathtt{v1}, \mathtt{v2} \leftarrow 0$
26: **else if** $\mathtt{k} = \tilde{\mathtt{k}}$ and $\mathtt{v1} \geq 0.4 \cdot \tilde{\mathtt{k}}$ **then**
27: rollback computation and transcript $\mathtt{T}$ to position $\mathtt{MP1}$ (update $N(\cdot)$ accordingly)
28: Reset Status: $\mathtt{k}, \mathtt{E}, \mathtt{v1}, \mathtt{v2} \leftarrow 0$
29: **else if** $\mathtt{k} = \tilde{\mathtt{k}}$ and $\mathtt{v2} \geq 0.4 \cdot \tilde{\mathtt{k}}$ **then**
30: rollback computation and transcript $\mathtt{T}$ to position $\mathtt{MP2}$ (update $N(\cdot)$ accordingly)
31: Reset Status: $\mathtt{k}, \mathtt{E}, \mathtt{v1}, \mathtt{v2} \leftarrow 0$
32: **else if** $\mathtt{k} = \tilde{\mathtt{k}}$ **then**
33: $\mathtt{v1}, \mathtt{v2} \leftarrow 0$
34: Output the outcome of Π corresponding to transcript $\mathtt{T}$

1) Modified Scheme: Let $\widetilde{\Pi}$ be the coding scheme described in Algorithm 1 that is obtained from [22, Algorithm 3] 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 $\Theta(\log(1/\epsilon))$. These changes imply that the length of the blocks $r = \sqrt{r_c/\epsilon}$ increases from $\Theta\left(\sqrt{1/\epsilon}\right)$ to $\Theta\left(\sqrt{\log(1/\epsilon)/\epsilon}\right)$, and the number of iterations R_{total} decreases to
$$\lceil n/r + 65\epsilon n \rceil = \Theta\left(\sqrt{\epsilon/\log(1/\epsilon)}\right)n + 65\epsilon n.$$
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 [22, Algorithm 3]. 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 this case the random string R at Line 5 can still be obtained by exchanging $\Theta\left(\sqrt{\epsilon \log(1/\epsilon)}\right)n$ 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 where 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 pad with zeros if the 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
$$\text{hash}_S(k),\ \widetilde{\text{hash}}_S(T),$$
$$\widetilde{\text{hash}}_S(T[1, \text{MP1}]),\ \widetilde{\text{hash}}_S(T[1, \text{MP2}]),$$
where the function hash_S is as defined in [22] 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 $h_S : \{0,1\}^{\leq m} \to \{0,1\}^o$ be the $\mathbb{F}_2$-linear hash function that satisfies, for every $x \in \{0,1\}^{\leq m}$ and $i \in [o]$, that
$$\left(h_S(x)\right)_i = \langle x, S[(i-1) \cdot m + 1, im] \rangle,$$
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
$$\widetilde{\text{hash}}_S : \{0,1\}^{\leq R_{\text{total}} \cdot r} \to \{0,1\}^{3o}$$
be the hash function that satisfies
$$\widetilde{\text{hash}}_S(T) = \Big(h_S(T), h_S(|T|), h_S(N'(T))\Big)$$
for every partial transcript $T \in \{0,1\}^{\leq R_{\text{total}} \cdot r}$ and its position string $N'(T)$.

2) Properties of the Modified Scheme: Next we state some properties of the coding scheme $\widetilde{\Pi}$ that are inherited from the protocol in [22]. We start with the following lemma which says that the simulation is successful as long as at most ϵn iterations suffer from a hash collision. This is the only proof we borrow without any modifications from [22], due to it being exactly the same for the original coding scheme Π and our slightly modified version $\widetilde{\Pi}$.

Lemma 11: 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 is instructive to summarize the correctness proof of the coding scheme from [22], which is the same as ours except for the hashes used: In order to prove correctness of the coding scheme Π of [22] a potential Φ is defined. [22, Lemmas 7.3 and 7.4] show that any iteration of the algorithm decreases this potential at most by a constant and every iteration in which no error or hash collision occurs increases the potential by at least one. The proof of the main theorem in [22], namely, Lemma 7.1, then shows that if at most $O(n\epsilon)$ hash collisions occur then the potential is so high that it a successful simulation is implied. [22, Lemmas 7.6, 7.7, and 7.8] show that for the various settings considered and their corresponding hashing schemes this bound of $O(n\epsilon)$ hash collisions is satisfied. These are the parts which we prove for our hashing scheme in detail below. Lemma 11 on the other hand exactly summarizes the part of the correctness proof of Π in [22] which applies, verbatim and without any modifications, to our coding scheme $\widetilde{\Pi}$, namely, that it works correctly as long as the hashing guarantees that no more than $O(n\epsilon)$ hash collisions occur.

We will also use the following claim which follows from Lemma 11 and states that if at most ϵn iterations suffer from a hash collision up to some iteration $t \in [R_{\text{total}}]$, then in most iterations $i \in [t]$ a new block was added to both T_A and T_B.

Claim 12: 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. Let $t \in [R_{\text{total}}]$ be some iteration and suppose that at

most ϵn iterations $i \in [t]$ in Γ suffer from a hash collision. Then there are at most $65\epsilon n$ iterations $i \in [t]$ in which no block was added to T_A and at most $65\epsilon n$ iterations $i \in [t]$ in which no block was added to T_B.

Proof: Suppose in contradiction that there are more than $65\epsilon n$ iterations $i \in [t]$ in which no block was added to T_A or more than $65\epsilon n$ iterations $i \in [t]$ in which no block was added to T_B. By symmetry we may assume that there are more than $65\epsilon n$ iterations $i \in [t]$ in which no block was added to T_A. To arrive at a contradiction we shall modify the string R to obtain a string R' such that when the string R in the run Γ is replaced with the string R' then on one hand, at most ϵn iterations in Γ suffer from a hash collision and on the other hand, the simulation performed by Γ is unsuccessful which contradicts Lemma 11 above.

Specifically, let $R' \in \{0,1\}^{R_{\text{total}} \cdot s}$ be the string which agrees with R on the first $t \cdot s$ bits and the last $(R_{\text{total}} - t) \cdot s$ bits are chosen such that no hash collision occurs after iteration t when the string R in the run Γ is replaced with the string R'. Such a choice exists since the output length of the hash functions is $o = c' \log(1/\epsilon)$ and the coding scheme is performing only a constant number of hash comparisons per iteration. Consequently, the probability that a uniform random seed in $\{0,1\}^s$ causes a hash collision at some iteration is at most $\exp(-\Omega(c' \log(1/\epsilon)))$, and in particular there exists a seed in $\{0,1\}^s$ that does not cause a hash collision at this iteration.

Let Γ' be the run of the coding scheme $\widetilde{\Pi}$ obtained from Γ by replacing the string R with the string R'. On one hand, we have that at most ϵn iterations in Γ' suffer from a hash collision. On the other hand, since Γ' and Γ behave the same on the first t iterations there are more than $65\epsilon n$ iterations in Γ' in which no block was added to T_A. But since $\widetilde{\Pi}$ is run for $n/r + 65\epsilon n$ iterations and since in each iteration at most one block is added to T_A, we have that at the end of the run Γ' less than n/r blocks of length r are present in T_A, and so the simulation is unsuccessful. This contradicts Lemma 11. ∎

B. Derandomization

In order to derandomize the coding scheme $\widetilde{\Pi}$ defined above we proceed according to the program outlined in the introduction. 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.

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 13 (Behavior pattern): Let Γ be a (possibly partial) run of the coding scheme $\widetilde{\Pi}$ (determined by the randomness string, the simulated noiseless 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 14 (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* $\mathrm{MP}b_A \neq \mathrm{MP}b'_B$ *but* $h_S(\mathrm{MP}b_A) = h_S(\mathrm{MP}b'_B)$.
7) *There exist* $b, b' \in \{1,2\}$ *such that* $N'(T_A[1, \mathrm{MP}b_A]) \neq N'(T_B[1, \mathrm{MP}b'_B])$ *but* $h_S(N'(T_A[1, \mathrm{MP}b_A])) = h_S(N'(T_B[1, \mathrm{MP}b'_B]))$.

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

Proof: By Condition 1 of Definition 14 we readily have that if $k_A \neq k_B$ then $\widetilde{\mathrm{hash}_S}(k_A) \neq \widetilde{\mathrm{hash}_S}(k_B)$. Next we show that if $T_A \neq T_B$ then $\widetilde{\mathrm{hash}_S}(T_A) \neq \widetilde{\mathrm{hash}_S}(T_B)$. If $|T_A| \neq |T_B|$ or $N'(T_A) \neq N'(T_B)$ then by Conditions 3 and 4 of Definition 14 we have that $\widetilde{\mathrm{hash}_S}(T_A) \neq \widetilde{\mathrm{hash}_S}(T_B)$. Otherwise, if $|T_A| = |T_B|$ and $N'(T_A) = N'(T_B)$, then we have that $T_A \oplus T_B = e$. Due to the linearity of h_S Condition 2 of Definition 14 implies that in this case $\widetilde{\mathrm{hash}_S}(T_A) \neq \widetilde{\mathrm{hash}_S}(T_B)$. A similar argument using Conditions 5, 6 and 7 of Definition 14 shows that if $T_A[1, \mathrm{MP}b_A] \neq T_B[1, \mathrm{MP}b'_B]$, for some $b, b' \in \{1,2\}$, then $\widetilde{\mathrm{hash}_S}(T_A[1, \mathrm{MP}b_A]) \neq \widetilde{\mathrm{hash}_S}(T_B[1, \mathrm{MP}b'_B])$. ∎

Finally, we show that hash mismatches depend only on the behavior pattern and the randomness string.

Claim 16: Given a string $R \in \{0,1\}^{R_{\mathrm{total}} \cdot s}$ *and a behavior pattern* $\mathcal{P}$ *of a (possibly partial) run* Γ *that uses the string* R *as the random string sampled at Line 5, one can efficiently determine the iterations in* Γ *in which a hash mismatch occurred. In particular, whether a hash mismatch occurred at some iteration in* Γ *depends entirely on the string* R *and the behavior pattern* $\mathcal{P}$.

Proof: By definition it holds that whether a hash mismatch occurred at some iteration in Γ depends only on the string R, the noise pattern in Γ and the values of the variables k, $|T|$, $N'(T)$, MP1 and MP2 for both parties at the beginning of this iteration. The noise pattern is included in the description of $\mathcal{P}$, and it can be verified by induction on the number of iterations that the values of the variables k, $|T|$, $N'(T)$, MP1 and MP2 for both parties depend only on the behavior pattern $\mathcal{P}$ and can be efficiently computed given $\mathcal{P}$. ∎

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 17 (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* $100\epsilon n$.

The following claim bounds the number of typical behavior patterns.

Claim 18: There are at most $2^{900\ H(\epsilon)n}$ *different typical behavior patterns.*

Proof: First note that there are at most $R_{\mathrm{total}} \leq n$ possible values for the number of iterations in $\mathcal{P}$, and that there are at most $R_{\mathrm{total}} \cdot (r + r_c) \leq 2n$ communication rounds in $\mathcal{P}$. Next observe that since the noise pattern has at most $2\epsilon n$ bit flips, then the number of different noise patterns is at most

$$\sum_{i=0}^{2\epsilon n} \binom{2n}{i} \leq 2\epsilon n \cdot \binom{2n}{2\epsilon n} \leq 2^{2H(\epsilon)n}.$$

Furthermore, since there are at most $100\epsilon n$ iterations in which no block was added to T_A, the number of different sets of such iterations is at most

$$\sum_{i=0}^{100\epsilon n} \binom{R_{\mathrm{total}}}{i} \leq 100\epsilon n \cdot \binom{n}{100\epsilon n} \leq 2^{100\ H(\epsilon)n}.$$

Finally, for each iteration in which no block was added to T_A we keep 3 bits of information and so the number of different possibilities for the values of these bits is at most $2^{300\epsilon n}$.

Concluding, we have that the number of different typical behavior patterns is at most

$$n \cdot 2^{2H(\epsilon)n} \cdot \left(2^{100H(\epsilon)n}\right)^2 \cdot \left(2^{300\epsilon n}\right)^2 \leq 2^{900H(\epsilon)n}.$$

∎

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

Claim 19: Let $\mathcal{P}$ *be any behavior pattern. According to Claim 16 one can determine how many iterations suffered from hash mismatches in a (partial) run with behavior pattern* $\mathcal{P}$ *and a given randomness string* R *by looking at* $\mathcal{P}$ *and* R *alone. If* R *is 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 such iterations, as determined by* $\mathcal{P}$ *and* R*, is at most* ϵn.

Proof: Suppose first that R is a uniform random binary string in $\{0,1\}^{R_{\mathrm{total}} \cdot s}$. In this case, since the output length of the hash functions is $o = c' \log(1/\epsilon)$ and since there are only constant number of conditions in Definition 14, the probability that a hash mismatch occurs at some iteration i is at most $2^{-\Omega(c' \log(1/\epsilon))}$. Consequently, the probability that more than ϵn iterations suffer from a hash mismatch is at most

$$\binom{n}{\epsilon n} \cdot 2^{-\Omega(c' \log(1/\epsilon)\epsilon n)} \leq 2^{-\Omega(c' \log(1/\epsilon)\epsilon n)},$$

where the inequality holds for sufficiently large constant c'.

In our case R is sampled from a δ-biased distribution for $\delta = 2^{-\Theta(no/r)}$ and consequently the probability that more than ϵn iterations suffer from a hash mismatch is at most

$$\begin{aligned} &2^{-\Omega(c'\epsilon \log(1/\epsilon)n)} + 2^{-\Theta(no/r)} \\ &\quad = 2^{-\Omega(c' \log(1/\epsilon)\epsilon n)} + 2^{-\Theta\left(\sqrt{\log(1/\epsilon)\epsilon n}\right)} \\ &\quad = 2^{-\Omega(c'\epsilon \log(1/\epsilon)n)}. \end{aligned}$$

∎

Claims 18 and 19 above imply the existence of a single random string R^* that leads to at most ϵn hash mismatches for all typical behavior patterns.

Corollary 20: For sufficiently large constant c'*, there is a string* $R^* \in \{0,1\}^{R_{\mathrm{total}} \cdot s}$ *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.

Finally, we show that when the coding scheme $\widetilde{\Pi}$ is run with the random string R^* guaranteed by the above corollary then the number of iterations suffering from *hash collisions* is at most ϵn.

Claim 21: 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.

Proof: If Γ has a typical behavior pattern then by our assumption we have that R^* leads to at most ϵn iterations in Γ suffering from hash mismatches. By Claim 15 this implies in turn that at most ϵn iterations in Γ suffer from a hash collision. Therefore it suffices to show that Γ has a typical behavior pattern.

Suppose in contradiction that Γ has a non-typical behavior pattern $\mathcal{P}$. Let $t \in [R_{\text{total}}]$ be the first iteration in Γ such that the number of iterations $i \in [t]$ in which no block was added to T_A is more than $65\epsilon n$ or the number of iterations $i \in [t]$ in which no block was added to T_B is more than $65\epsilon n$. Let $\mathcal{P}'$ be the (partial) behavior pattern obtained by restricting $\mathcal{P}$ to the first t iterations. Then $\mathcal{P}'$ is a typical behavior pattern and consequently by our assumption we have that the number of iterations suffering from hash mismatches determined by $\mathcal{P}'$ and R^* is at most ϵn. Furthermore, since $\mathcal{P}$ and $\mathcal{P}'$ agree on the first t iterations we have that the number of iterations suffering from hash mismatches determined by $\mathcal{P}$ and R^* among the first t iterations is at most ϵn. By Claim 15 this implies in turn that at most ϵn iterations $i \in [t]$ in Γ suffer from a hash collision which contradicts Claim 12. ∎

3) Completing the Proof of Lemma 5: We are now ready to complete the proof of the main result in this section.

Proof of Lemma 5: Corollary 20 and Claim 21 guarantee the existence of a string $R^* \in \{0,1\}^{R_{\text{total}} \cdot s}$ such that in any run of the coding scheme $\widetilde{\Pi}$ that uses the string R^* as the random string sampled at Line 5 and has at most ϵ fraction of adversarial errors, the number of iterations suffering from a hash collision is at most ϵn. By Lemma 11 this implies in turn that any run of the coding scheme $\widetilde{\Pi}$ that uses R^* as the random string sampled at Line 5 and has at most ϵ fraction of adversarial errors successfully simulates the noiseless protocol π.

To show that $\widetilde{\Pi}$ has the required rate note that the total number of bits communicated during the execution of $\widetilde{\Pi}$ is

$$\begin{aligned} R_{\text{total}} \cdot \big(r + r_c\big) &= \left(\tfrac{n}{r} + \Theta(n\epsilon)\right) \cdot r \cdot \left(1 + \tfrac{r_c}{r}\right) \\ &= n \cdot \left(1 + \Theta(r\epsilon)\right) \cdot \left(1 + \tfrac{r_c}{r}\right) \\ &= n \cdot \left(1 + \Theta\left(r\epsilon + \tfrac{r_c}{r}\right)\right). \end{aligned}$$

Due to our choice of $r = \Theta\big(\sqrt{\log(1/\epsilon)/\epsilon}\big)$ and $r_c = \Theta(\log\ (1/\epsilon))$ the above implies in turn that the number of bits communicated in the coding scheme $\widetilde{\Pi}$ is

$$n \cdot \big(1 + \Theta(\sqrt{\epsilon \log\ (1/\epsilon)})\big).$$

So the rate of $\widetilde{\Pi}$ is

$$1 - O\big(\sqrt{\epsilon \log\ (1/\epsilon)}\big) = 1 - O\big(\sqrt{H(\epsilon)}\big).$$

Finally, observe that one can find the string R^* by going over all pairs $(\mathcal{P}, R)$ where $\mathcal{P}$ is a typical behavior pattern and $R \in \{0,1\}^{R_{\text{total}} \cdot s}$ is in the support of the δ-biased distribution for $\delta = 2^{-\Theta(no/r)}$ which requires $\Theta\big(\sqrt{\epsilon \log(1/\epsilon)}\big)n$ random bits. Therefore, the number of possible strings R is at most $2^{O(n)}$. Furthermore, by Claim 18 there are at most $2^{O(n)}$ different typical behavior patterns $\mathcal{P}$. Therefore the total number of pairs $(\mathcal{P}, R)$ one needs to check is at most $2^{O(n)}$. Finally, Claim 16 shows that for each such pair it takes $\text{poly}(n)$ time to verify whether the number of iterations suffering from a hash mismatch determined by this pair is at most ϵn, and we conclude that the total time this process takes is at most $2^{O(n)}$. ∎

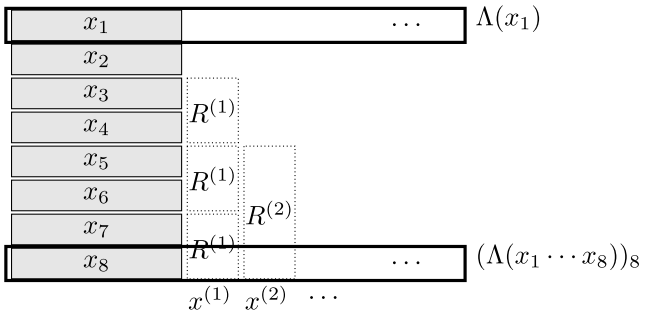

Fig. 1. An illustration of the tree code's encoding.

VI. The Outer Code: Proof of Lemma 6

A. The Tree Code Construction

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.

In more detail, the main ingredient in our tree code construction is the following lemma showing 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 that for any integers k, n that satisfy $\Omega((\log n)/\epsilon) \le k \le n$, there exists 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 Facts 2 and 3, and we defer its proof to Section VI-D.

Lemma 22: 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 a fraction $\delta'/2$ of errors in time $\text{poly}(n)$.

The construction of the tree code Λ is 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 $x \in \Sigma_{\text{in}}^n$ is 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 Lemma 22 for a 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 then 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.

B. Distance

The distance guarantee of the tree code 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 $C^{(t)}$.

Claim 23: 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. For any $j \in [i, n]$ it holds that

$$\text{dist}\Big(\Lambda(x)[i,j], \Lambda(y)[i,j]\Big) \geq \min\left\{\frac{\epsilon}{2\ k_0 \log n}, \frac{1}{16(\log n)/\epsilon + 8}\right\}.$$

Lemma 6 then holds as a corollary of the above claim by setting $\delta_0 := 1/(32\ k_0)$.

Proof: If $j - i < 2m$ then

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

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 the 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 the maximality of t.

Note that $\Lambda(x)[i_0+1, i_0+2^t]$ contains $x[i_0+1, i_0+2^t]$ as the systematic parts of $C^{(t)}(x[i_0+1, i_0+2^t])$. Also note that by our construction, $\Lambda\ (x)[i_0+1+2^t, i_0+2\cdot 2^t]$ contains the redundant part $R^{(t)}(x[i_0+1, i_0+2^t])$ of $C^{(t)}(x[i_0+1, i_0+2^t])$. In a symmetric way, the same holds for $\Lambda(y)$ and y.

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 $C^{(t)}$ (as given by Lemma 22) we have that

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

Equation (2) implies 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 we get that

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

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 $j - i < 4 \cdot 2^t$. Recall that $\delta' = \frac{1}{2(\log n)/\epsilon+1}$ to complete the proof. ∎

C. 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 for which a codeword $C^{(t)}$ is fully contained 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 $2\ m$, in which case we simply set y in this interval as the systematic part of w in the corresponding interval.

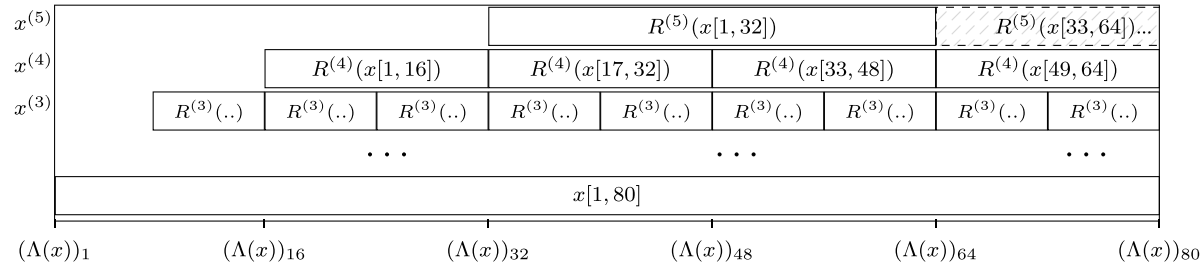

Fig. 2. An illustration of the first 80 indices of $\Lambda(x)$, the encoding of $x \in \Sigma_{\text{in}}^n$ using our tree code.

The formal description of the decoding procedure follows.

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

0) Initialize: $\ell := 1$ // Left index of current interval
1) If $j - \ell < 2\,m$, set $y[\ell, j]$ to be the systematic part of $w[\ell, j]$ and output y.
2) Otherwise, let t be the 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 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 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 $x \in \Sigma_{\text{in}}^j$ is such that $\text{dist}_{\text{sfx}}(\Lambda(x), w) \leq \frac{\delta_0 \cdot \epsilon}{2 \log n}$. 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

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

where the equality follows due to our choice of $m = k_0(\log n)/\epsilon$ and $\delta_0 = 1/(32\ k_0)$. 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

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

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

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

and similarly

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

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 $y[\ell, \ell - 1 + 2^t] = x[\ell, \ell - 1 + 2^t]$.

D. Proof of Lemma 22

We now complete the proof of Lemma 6 by proving Lemma 22. Lemma 22 follows by substituting $\rho = 1/2$, $s = (\log n)/\epsilon$ and $r = \log n$ in the following lemma which shows the existence of a systematic error-correcting code with good rate and distance.

Lemma 24: For every $0 < \rho < 1$ *there exist* $\delta > 0$ *and integer* $k_0 \in \mathbb{N}$ *such that the following holds for any integers* $k, s, r \in \mathbb{N}$ *that satisfy* $k \cdot \frac{\rho r}{s} \geq k_0$ *and* $s \geq \log(k(1 + \frac{\rho r}{s}))$. *There exists a systematic* $\mathbb{F}_2$*-linear code* $C : \Sigma_{\text{in}}^k \to \Sigma_{\text{out}}^k$ *with* $\Sigma_{\text{in}} = \{0, 1\}^s$, $\Sigma_{\text{out}} = \{0, 1\}^{s+r}$, *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 Fact 3 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 assume $k \cdot \frac{\rho r}{s} \geq k_0$). Let RS be the Reed-Solomon code guaranteed by Fact 2 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.

Next 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

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

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. ∎

REFERENCES

[1] S. Agrawal, R. Gelles, and A. Sahai, "Adaptive protocols for interactive communication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 595–599.

[2] N. Alon, M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler, "Reliable communication over highly connected noisy networks," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2016, pp. 165–173.

[3] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Struct. Algorithms*, vol. 3, no. 3, pp. 289–304, 1992.

[4] Z. Brakerski, Y. T. Kalai, and M. Naor, "Fast interactive coding against adversarial noise," *J. ACM*, vol. 61, no. 6, pp. 35:1–35:30, Dec. 2014.

[5] M. Braverman, "Towards deterministic tree code constructions," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, 2012, pp. 161–167.

[6] M. Braverman and K. Efremenko, "List and unique coding for interactive communication in the presence of adversarial noise," *SIAM J. Comput.*, vol. 46, no. 1, pp. 388–428, 2017.

[7] M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler, "Constant-rate coding for multiparty interactive communication is impossible," *J. ACM*, vol. 65, no. 1, 2017, Art. no. 4.

[8] M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky, "Coding for interactive communication correcting insertions and deletions," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6256–6270, Oct. 2017.

[9] M. Braverman and A. Rao, "Toward coding for maximum errors in interactive communication," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 7248–7255, Nov. 2014.

[10] K. Censor-Hillel, R. Gelles, and B. Haeupler, "Making asynchronous distributed computations robust to channel noise," in *Proc. 9th Innov. Theor. Comput. Sci. Conf. (ITCS)*, vol. 94. 2018, pp. 50:1–50:20. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/8318, doi: 10.4230/LIPIcs.ITCS.2018.50.

[11] G. Cohen, B. Haeupler, and L. Schulman, "Explicit binary tree codes with polylogarithmic size alphabet," in *Proc. 50th Annu. ACM Symp. Theory Comput.*, 2018, pp. 1–25.

[12] U. Feige and J. Kilian, "Finding OR in a noisy broadcast network," *Inf. Process. Lett.*, vol. 73, nos. 1–2, pp. 69–75, 2000.

[13] G. D. Forney, Jr., "Concatenated codes," Massachusetts Inst. Technol., Res. Lab. Electron., Cambridge, MA, USA, Tech. Rep. 440, 1965.

[14] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman, "Optimal coding for streaming authentication and interactive communication," *IEEE Trans. Inf. Theory*, vol. 61, no. 1, pp. 133–145, Jan. 2015.

[15] R. Gelles, "Coding for interactive communication: A survey," *Found. Trends Theor. Comput. Sci.*, vol. 13, nos. 1–2, pp. 1–157, 2017.

[16] R. Gelles, A. Moitra, and A. Sahai, "Efficient coding for interactive communication," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1899–1913, Mar. 2014.

[17] R. Gelles and Y. T. Kalai, "Constant-rate interactive coding is impossible, even in constant-degree networks," in *Proc. Leibniz Int. Inform.*, vol. 67. 2017, pp. 1–13.

[18] M. Ghaffari and B. Haeupler, "Optimal error rates for interactive coding II: Efficiency and list decoding," in *Proc. IEEE 55th Symp. Found. Comput. Sci.*, Oct. 2014, pp. 394–403.

[19] M. Ghaffari, B. Haeupler, and M. Sudan, "Optimal error rates for interactive coding I: Adaptivity and other settings," in *Proc. 46th Annu. ACM Symp. Theory Comput.*, 2014, pp. 794–803.

[20] E. N. Gilbert, "A comparison of signalling alphabets," *Bell Syst. Tech. J.*, vol. 31, no. 3, pp. 504–522, 1952.

[21] B. Haeupler, "Interactive channel capacity revisited," in *Proc. IEEE 55th Symp. Found. Comput. Sci.*, Oct. 2014, pp. 226–235.

[22] B. Haeupler. (Nov. 2014). "Interactive channel capacity revisited." [Online]. Available: https://arxiv.org/abs/1408.1467

[23] B. Haeupler and A. Shahrasbi, "Synchronization strings: Codes for insertions and deletions approaching the Singleton bound," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput.*, 2017, pp. 33–46.

[24] B. Haeupler, A. Shahrasbi, and E. Vitercik. (2017). "Synchronization strings: Channel simulations and interactive coding for insertions and deletions." [Online]. Available: https://arxiv.org/abs/1707.04233

[25] B. Haeupler and A. Velingker, "Bridging the capacity gap between interactive and one-way communication," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2017, pp. 2123–2142.

[26] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[27] G. Kol and R. Raz, "Interactive channel capacity," in *Proc. 45th Annu. ACM Symp. Theory Comput.*, 2013, pp. 715–724.

[28] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," *SIAM J. Comput.*, vol. 22, no. 4, pp. 838–856, Aug. 1993.

[29] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[30] L. J. Schulman, "Communication on noisy channels: A coding theorem for computation," in *Proc. IEEE Symp. Found. Comput. Sci., Annu.*, Oct. 1992, pp. 724–733.

[31] L. J. Schulman, "Deterministic coding for interactive communication," in *Proc. 25th Annu. ACM Symp. Theory Comput.*, 1993, pp. 747–756.

[32] L. J. Schulman, "Coding for interactive communication," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1745–1756, Nov. 1996.

[33] L. J. Schulman. (2003). *A Postcript to 'Coding for Interactive Communication'*. [Online]. Available: http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt

[34] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

[35] R. R. Varshamov, "Estimate of the number of signals in error correcting codes," *Doklady Akademii Nauk SSSR*, vol. 117, no. 5, pp. 739–741, 1957.

[36] A. C.-C. Yao, "Some complexity questions related to distributive computing(preliminary report)," in *Proc. 11th Annu. ACM Symp. Theory Comput.*, 1979, pp. 209–213.

Authors' biographies not available at the time of publication.