Palette-Alternating Tree Codes

Gil Cohen
Department of Computer Science, Tel Aviv University, Israel
gil@tauex.tau.ac.il

Shahar Samocha
Department of Computer Science, Tel Aviv University, Israel
samocha@mail.tau.ac.il

Abstract

A tree code is an edge-coloring of the complete infinite binary tree such that every two nodes of equal depth have a fraction–bounded away from 0–of mismatched colors between the corresponding paths to their least common ancestor. Tree codes were introduced in a seminal work by Schulman [29] and serve as a key ingredient in almost all deterministic interactive coding schemes. The number of colors effects the coding scheme's rate.

It is shown that 4 is precisely the least number of colors for which tree codes exist. Thus, tree-code-based coding schemes cannot achieve rate larger than 1/2. To overcome this barrier, a relaxed notion called *palette-alternating tree codes* is introduced, in which the number of colors can depend on the layer. We prove the existence of such constructs in which most layers use 2 colors–the bare minimum. The distance-rate tradeoff we obtain matches the Gilbert-Varshamov bound.

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity. To analyze our protocol, we prove a structural result on the location of failed communication-rounds induced by the error pattern enforced by the adversary. Our coding scheme is efficient given an explicit palette-alternating tree code and serves as an alternative to the scheme obtained by [13].

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Tree Codes, Coding Theory, Interactive Coding Scheme

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.11

Funding The research leading to these results has received funding from the Israel Science Foundation (grant number 1569/18) and from the Azrieli Faculty Fellowship.

Acknowledgements We wish to thank Leonard J. Schulman for many insightful discussions over the years regarding tree codes and interactive coding schemes.

1 Introduction

Tree codes are a powerful combinatorial structure, defined and proven to exist in [29] in order to serve as a key ingredient for achieving a constant rate interactive coding scheme. Tree codes are the central object for encoding information in the interactive coding theory which developed from the initial papers. They remain a crucial building block in almost all interactive coding schemes [26, 10, 8, 13, 3, 5, 2, 4, 16, 17, 22, 1, 14, 7, 19, 32].

We turn to formally define tree codes. Let T be a rooted binary tree that is endowed with an edge coloring from some ambient color set (or alphabet) Σ. Let u, v be a pair of vertices in T with equal depth and a least common ancestor w. Let ℓ be the distance, in edges, from u to w. Let $p_u, p_v \in \Sigma^\ell$ be the sequences of colors on the path from w to u and to v, respectively. We define $h(u, v)$ to be the relative Hamming distance between p_u and p_v.

▶ **Definition 1** (Tree codes [29]). *Let T be the complete rooted infinite binary tree. The tree T, together with an edge-coloring of T by a color set Σ is called a* tree code with distance δ *if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. When there is no $\delta > 0$ for which T is a tree code with distance δ we say that T has* vanishing distance.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 11; pp. 11:1–11:29
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Schulman [29] proved that for every distance parameter $\delta < 1$ tree codes with a constant number of colors $c = c(\delta)$ exist. Although tree codes are used in different ways by different interactive coding schemes, one aspect is common to all: When a party wishes to send a bit, a suitable color from Σ is sent instead. Thus, the rate of all tree-code-based coding schemes is bounded above by $1/\log_2|\Sigma|$. One is led to ask a natural combinatorial question–what is the least number of colors in a tree code with non-vanishing distance?

1.1 Tree codes: 4 colors suffice and are necessary

We first observe that 3 colors do not suffice and, as a result, the rate of every tree-code-based coding scheme cannot exceed 1/2, let alone approach capacity. Consider any 3-color tree code. First, we may assume that every two siblings are connected to their parent with edges having distinct colors as otherwise the distance of the tree code is 0. Let u, v be any two vertices. Out of u, v go four edges and so by the pigeonhole principal in every 3-coloring, two of these edges share the same color. By the above, one of these edges goes out of u and the other goes out of v. This implies that, starting from the two sons of the root, one can construct two paths of any desired length $n \geq 1$ with the same color pattern, establishing that the tree has vanishing distance.

Based on the ideas Schulman introduced to prove the existence of tree codes with a constant number of colors, we complement the above observation and establish that 4 colors suffice for a tree code with non-vanishing distance.

▶ **Theorem 2.** *There exists a* 4*-color tree code with distance* 0.136*.*

The proof of Theorem 2 appears in Section 3. As Schulman's original proof for the existence of tree codes, Theorem 2 is nonconstructive. Coming up with explicit constructions of non-vanishing distance tree codes with a constant number of colors is one of the most challenging problems in this field [30, 15, 6, 25, 23, 13, 11, 24]. The currently best known result [11] guarantees any designated distance $\delta < 1$ when using $(\log n)^{O_\delta(1)}$ colors at depth n. This work, however, concerns with the information-theoretic aspect of the channel capacity, and the computational aspects are left for future work.

While our proof of Theorem 2 closely follows Schulman's proof, and the observation that 4 colors are necessary is easy to prove, to the best of our knowledge, this basic combinatorial result was not known and, furthermore, we find it surprising that merely 4 colors suffice to guarantee such a strong combinatorial structure. Still, even if 4 is a surprisingly small number of colors, an interactive coding scheme that uses a 4-color tree code would have rate bounded above by 1/2.

1.2 Palette-alternating tree codes

To save on communication, one might hope to avoid the use of the tree code "every now and then". However, if one sends a bit in the clear without encoding it, and that bit is flipped by the adversary, the simulation seems doomed to fail without some way of generating an unpredictable verification (which can be done when considering randomized schemes). Perhaps a better idea would be to try and apply puncturing–a standard tool from classic error correcting codes used for improving the rate of a code. However, the distance of a tree code is far more sensitive than the distance of a standard error correcting code. In particular, changing the color of a single edge can cause the distance to vanish. It is thus not clear how one can "puncture" a tree code without vanish its distance.

Our key insight is to consider a variant of tree codes we call *palette-alternating tree codes* in which the number of colors is allowed to depend on the depth. A good first example to have in mind is a coloring that uses 4 colors in even layers and 2 colors in odd layers. To our surprise, such palette-alternating tree codes with non-vanishing distance exist! To calculate the rate-overhead incurred by using this palette-alternating tree code, observe that the number of bits sent when using an (even) depth-n palette-alternating tree code is

$$\frac{n}{2}\log_2 2 + \frac{n}{2}\log_2 4 = \frac{3}{2}n,$$

and so the rate incurred by the encoding is $2/3$, improving upon the $1/2$ rate one would get by using the best available tree code. Note that this even beats the rate of a 3-color tree code–had it existed–since $\log_2 3 > 3/2$. Put differently, in an amortized sense, the palette-alternating tree code above requires only $2^{3/2} \approx 2.83$ colors.

One can get greedy and ask whether a palette-alternating tree code that uses, say, 4 colors at layers $0, 3, 6, ...$ and 2 colors in the remaining layers exist. If so, one can potentially improve the scheme's rate to $3/4$. We prove the existence of such palette-alternating tree codes. In fact, we show that one can use 4 colors as seldom as she please and 2 colors–the bare minimum–in most layers. We turn to give a formal treatment of the above discussion.

▶ **Definition 3** (Palette-alternating tree codes). *Let $\Sigma_0, \ldots, \Sigma_{c-1}$ be (not necessarily distinct) sets. Let T be the complete rooted infinite binary tree. A* palette-alternating tree code *is an edge-coloring of T where at layer $t \in \mathbb{N}$ the colors are taken from the set $\Sigma_{t \pmod c}$. T is said to have distance δ if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. We define the* rate *ρ of T to be the number satisfying*

$$\frac{1}{\rho} = \frac{1}{c}\sum_{i=0}^{c-1}\log_2 |\Sigma_i|.$$

We suggest that the flexibility introduced by palette-alternating tree codes allows one to better capture the notion of rate in the online setting. Indeed, the importance of rate is only significant when "long" messages are being sent and so, informally, using a big palette of colors only once in a while should not be considered as an indication of poor rate. Our definition of rate formalizes that property. Note that we still insist on having the distance measured in terms of worst-case–a must as we wish to replace tree codes with palette-alternating tree codes in interactive coding schemes. It is only the rate that is being, in a sense, amortized.

As mentioned, we prove that palette-alternating tree codes can have rate approaching arbitrarily close to 1 while maintaining non-vanishing distance, thus bypass the $1/2$ bound proven for (standard) tree codes.

▶ **Theorem 4.** *For every $\varepsilon > 0$ there exists a palette-alternating tree code with rate $1 - \varepsilon$ and distance $\delta = \Omega(\varepsilon \cdot \log^{-1}(1/\varepsilon))$.*

Comparison with the Gilbert-Varshamov bound

Observe that the distance-rate trade-off obtained in Theorem 4 is the same as the one obtained by the Gilbert-Varshamov bound for standard offline binary error correcting codes, and in particular is optimal (up to constant factors). Interestingly, while it is known that the channel capacity in the online setting is $1 - \Theta(\sqrt{\varepsilon \log(1/\varepsilon)})$–significantly lower than in the offline setting [20], the online requirement on the encoding function itself does not cost more in terms of the distance-rate trade-off. Rather, it is the additional overhead incurred by the mechanism required for synchronization that is responsible for the lower channel capacity in the online setting. We elaborate more on this in Section 1.3.

The proof of Theorem 4, which can be found in Section 4, is based on a variant of the construction we use in Theorem 2. There, the alphabet symbols are taken from the field of four elements, $\mathbb{F}_4$. The key idea in obtaining the savings in the alphabet size is to trace the $\mathbb{F}_4$ field elements down to $\mathbb{F}_2$ in most layers. Interestingly, we cannot afford to work over the field $\mathbb{F}_3$ as we crucially rely on the fact that the characteristic of the fields is 2 as well as on the smaller field being a subfield of the larger one.

1.2.1 Palette-alternating tree codes: further discussion and generalization

We remark that it is not clear if one can start from an arbitrary 4-color tree code and change some of the layers to have only 2 colors (in a sense, effectively puncturing the 4-color tree code) while maintaining non-vanishing distance. Our proof seems to have the effect of "correlating" the colors in the 4-color layers with the paths that contain them. To emphasize this point, note that a 2-color layer does not immediately "buy" us redundancy. Nevertheless, the 2-color layers have the important task of making sure that the 4-color layers do. Indeed, by switching the colors of siblings in the 2-color layers one can potentially vanish the distance.

It is also interesting to compare palette-alternating tree codes that use 2 colors in most layers with some of the known probabilistic schemes [20, 18] that take the following strategy: in most rounds simulate the protocol as is (namely, assuming no errors occur) and only rarely verify the transcript using hash functions. It is tempting to compare the 2-color layers in a palette-alternating tree code with the error-free part of the simulation and the 4-color layers with the verification rounds. Indeed, at the very least, both the 2-color layers and the error-free part cost nothing in terms of rate. The crucial difference, however, lies in the fact that while the error-free simulation does not carry any weight in terms of error correction, the 2-color layers do.

We end this section by proposing a more general, and arguable more natural, definition than palette-alternating tree codes which allows for different palettes used at different layers without being necessarily periodical. While our proof of Theorem 4 yields a palette-alternating tree code, we believe that the more general definition is worth presenting here. For simplicity, we identify a finite color set Σ with $\{1, 2, \ldots, |\Sigma|\}$.

▶ **Definition 5** (Dynamic-Palette Tree Codes). *Let $c : \mathbb{N} \to \mathbb{N}$. Let T be the complete rooted infinite binary tree. A* dynamic-palette tree code *is an edge-coloring of T where at layer $t \in \mathbb{N}$ the colors are taken from the set $\{1, 2, \ldots, c(t)\}$. T is said to have distance δ if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. We define the rate ρ of T to be the number satisfying*

$$\frac{1}{\rho} = \inf_{\ell \in \mathbb{N}} \frac{1}{\ell} \sum_{i=1}^{\ell} \log_2 c(i).$$

1.3 Interactive coding schemes

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity. Our coding scheme is efficient given an explicit construction of palette-alternating tree codes and serves as an alternative to the scheme obtained by Gelles *et al.* [13]. In this section we describe our result and proof technique. We start by reviewing basic notions in interactive coding schemes.

Communication complexity

Communication complexity addresses a basic question: If several parties wish to compute a function of the information they jointly possess, how long does their conversation need to be? In its most basic form, one considers two parties, Alice and Bob, that would like to jointly compute a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ of their respective inputs $x, y \in \{0,1\}^n$. The parties can communicate over a channel, and their goal is to compute $f(x,y)$ by exchanging as few bits as possible.

An interactive computation as above is performed via a communication protocol π which consists of a pair of algorithms π_A and π_B run by Alice and Bob, respectively. In this paper we focus on deterministic protocols, that is, π_A and π_B are deterministic algorithms. Informally, the communication is performed in rounds where the protocol dictates what is sent in each round based on the round number, the input of the party, and the bits received so far. After some number of rounds $r = r(x,y)$ the protocol terminates, at which point both parties know $f(x,y)$. The *(deterministic) communication complexity* of the protocol π is given by $\mathsf{CC}(\pi) = \max_{x,y} r(x,y)$. The *communication complexity* of f, denoted by $\mathsf{CC}(f)$, is the minimum of $\mathsf{CC}(\pi)$ over all protocols π that compute f.

Interactive coding schemes

One aspect that is always an issue when considering communication are errors in transmission introduced by imperfect or compromised channels. The research field of coding for interactive communication that addresses this issue was initiated in a sequence of seminal papers by Schulman [28, 29, 31], and is by now an active and exciting research field (see Gelles's excellent survey [12]). There are several models one can consider. For examples, transmitted bits can be erased (replaced with a senseless symbol $\perp$) or worse–flipped–leaving no trace to the occurred error. In this paper we focus on perhaps the most well-studied model in which bits can be flipped. Further, we consider the most difficult setting of *adversarial errors* in which any ε-fraction of the bits might be flipped.

A protocol π is said to be *ε-resilient* if the protocol preserves its functionality even at the presence of ε-fraction of adversarial errors. The *ε-resilient communication complexity* of f, denoted by $\mathsf{CC}_\varepsilon(f)$, is the minimum of $\mathsf{CC}(\pi)$ over all ε-resilient protocols π that compute f. For any fixed function f it is clear that $\mathsf{CC}_\varepsilon(f)$ is non-decreasing as ε increases. In the extreme cases $\mathsf{CC}_0(f) = \mathsf{CC}(f)$ whereas $\mathsf{CC}_1(f) = \infty$, namely, $\mathsf{CC}_1(f)$ is unbounded.

Resilient protocols are typically obtained by devising an *interactive coding scheme* which, informally, is a compiler CS_ε that is parameterized by the resiliency parameter ε. Given a protocol π, the interactive coding scheme produces an ε-resilient protocol $\mathsf{CS}_\varepsilon(\pi) = \pi_\varepsilon$ that computes the same function as π. The goal is to design an interactive coding scheme with low overhead in communication. Namely, one would like to maximize $\rho(\pi) = \mathsf{CC}(\pi)/\mathsf{CC}(\pi_\varepsilon)$. The *rate* of the interactive coding scheme $\rho(\mathsf{CS}_\varepsilon)$ is the infimum of $\rho(\pi)$ over all protocols π.

Channel capacity

Focusing on the channel itself, rather than on any specific function f, one can define the *channel capacity* $\mathsf{Cap} : [0,1] \to [0,1]$ by

$$\mathsf{Cap}(\varepsilon) = \inf_f \left(\frac{\mathsf{CC}(f)}{\mathsf{CC}_\varepsilon(f)} \right),$$

where the infimum is taken over all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ for all $n \geq 1$. Note that $\mathsf{Cap}(0) = 1$ whereas $\mathsf{Cap}(1) = 0$. A fundamental problem in interactive coding theory, and the focus of this work, is the study of the channel capacity $\mathsf{Cap}(\varepsilon)$.

We remark that the channel capacity can be defined with respect to other models and a huge body of work is devoted to the study of the channel capacity in our setting as well as for other channels, most notably binary symmetric channels (BSC) in which every bit is flipped independently with probability ε. Moreover, one needs to specify other properties of the protocols so as to formalize the problem. For example, is the turn of speak predetermined by the protocol or can it depend on the exchanged bits? In case of such "adaptive" protocols, what happens if both parties send a message at the same round?

As in most works, we focus on non-adaptive protocols in which the turn of speak is fixed in advance. For concreteness, we focus on *alternating protocols* where Alice speaks at even rounds and Bob speaks at odd rounds. We made this choice mostly for convenience and our results can be straightforwardly generalized. We also assume that the channel is binary. This is the most difficult setting and allowing for channels over a larger alphabet, especially one that can depend on the error parameter ε, only makes the problem of devising protocols easier.

In his seminal work [29], Schulman proved that $\mathsf{Cap}(\varepsilon) > 0$ for some $\varepsilon > 0$. In a tour de force result, Kol and Raz [20] gave a tight bound of $\mathsf{Cap}(\varepsilon) = 1 - \Theta(\sqrt{\varepsilon \log 1/\varepsilon})$ on the channel capacity in this setting for non-adaptive probabilistic protocols. Their upper bound clearly holds for adversarial errors as well. Gelles *et al.* [13] gave the first deterministic coding scheme against adversarial errors, derandomizing Haeupler's protocol [18], that approaches capacity, namely, their coding scheme has rate $1 - O(\sqrt{\varepsilon \log 1/\varepsilon})$.

1.4 Capacity approaching coding schemes via palette-alternating tree codes

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity and thus matches the rate obtained by [13]. The advantage of our coding scheme is that given an explicit construction of palette-alternating tree codes, our scheme is efficient. We believe that the recent progress on tree code constructions [11, 24] may eventually lead to constructions of palette-alternating tree codes. The coding scheme suggested in [13], on the other hand, relies on a certain counting argument, and it is not clear to us how to obtain an efficient scheme based on these ideas.

▶ **Theorem 6.** *Let $\varepsilon > 0$. Assume there exists an explicit palette-alternating tree code with rate $1 - \varepsilon$ and distance $\delta = \Omega(\varepsilon \cdot \log^{-1}(1/\varepsilon))$ (which, computational aspects aside, we know exists by Theorem 4). Then, there exists an efficient deterministic coding scheme against ε-fraction of adversarial errors with rate $1 - O(\sqrt{\varepsilon \log(1/\varepsilon)})$.*

1.4.1 Proof idea

In the remaining of this section, we elaborate on some of the ideas that go into our construction and analysis of Theorem 6.

1.4.1.1 Synchronization

Interactive coding schemes that make use of tree codes do not simply encode the bits that are meant to be sent by the non-resilient protocol π using the tree code. These schemes also need to implement a mechanism for making sure that both parties are, in a sense, synchronized. Indeed, informally, the errors have the effect of causing the parties to transmit data with respect to information that was never sent to them. Without a way to synchronize, even with no additional errors, the parties will not be able to make progress on simulating the protocol as the information they exchange is irrelevant.

Thus, on top of the bits that the parties would have communicate without the presence of errors, some meta data used for synchronization must be maintained and transmitted. Both the "data bits" as well as the "sync bits" are encoded using a tree code before sent over the channel. Thus, the rate of deterministic interactive coding schemes is determined both by the rate of the tree code as well as by the overhead required for synchronization.

To obtain interactive coding schemes with rate approaching 1 we need, on top of replacing a tree code with a palette-alternating tree code, to have a low overhead in synchronization. There are two main obstacles for accomplishing that:

1. One must argue that not too many sync bits are needed to successfully maintain synchronization; and
2. One needs to distinguish between sync bits and data bits which in previous works was effectively done by sending a bit indicating the bit "type" (more precisely, a larger alphabet was used followed by an alphabet reduction).

The first issue is fairly straightforward to handle. Indeed, it is intuitive that in a sensible scheme, the amount of synchronization required is proportional to the fraction of errors and this is true for both Schulman's coding scheme [29] and for Braverman-Rao's scheme [9]. The second issue requires more care. Braverman-Rao's scheme is very dynamic and on any given round the bit type depends on the error pattern enforced by the adversary. Although most bits are data bits, it seems difficult to argue that their scheme can be made to have high rate. Luckily, we are able to devise a coding scheme based on some adaptation of Schulman's original ideas. The coding scheme obtained, however, does not approach capacity, and has rate $1 - \tilde{O}(\sqrt[3]{\varepsilon})$ (see Section 5.3). Further ideas are required to prove Theorem 6 which we discuss next.

1.4.1.2 Clusters of failed decoding rounds

In order to approach capacity, we examine more closely the effect that adversarial errors have on (palette-alternating) tree codes. Schulman's analysis is based on bounding the number of rounds in which decoding fails. More precisely, it was shown [29] that if one encodes using a tree code with distance $\delta_{\mathcal{TC}}$ then at most $O(\varepsilon/\delta_{\mathcal{TC}})$ fraction of rounds would result in failed decoding. We prove a structural result, refining the quantitative one, regarding *where* these "bad" rounds may occur as a function of the locations of the adversarial errors. We show that the bad rounds are, in a sense, clustered around the errors that are introduced. We exploit this structure to obtain a tighter analysis of our protocol, and achieve the stated, optimal, rate.

1.5 Organization

In Section 2 we give the formal definitions of protocols and interactive coding schemes, as well as setting notation and state some known results we use. In Section 3 we prove Theorem 2 which asserts that 4-color tree codes exist. While not directly applicable to our proof of Theorem 6, we encourage the reader to read the proof (including Section 3.1) as ideas from the proof will be used for proving the existence of palette-alternating tree codes (Theorem 4). In Section 4 we prove Theorem 4. Lastly, in Section 5, we prove Theorem 6 where first, in Section 5.3, we give a sub-optimal analysis.

2 Preliminaries

Unless otherwise stated, all logarithms are taken to the base 2. We denote by $\mathbb{N}$ the set of natural numbers (of course, including 0), and write $\mathbb{N}_1$ for $\mathbb{N} \setminus \{0\}$. For integers $a \leq b$ we write $[a, b]$ for all integers in this interval. For an integer $c \geq 1$, we let $[c] = \{1, 2, \ldots, c\}$. We follow the convention that strings are indexed starting from 1. For two strings $x, y \in \Sigma_1 \times \cdots \times \Sigma_n$, we denote by $\Delta(x, y)$ their hamming distance. We make use of the following standard inequalities.

▶ **Lemma 7.** *For every integers $1 \leq k \leq n$ with $\frac{k}{n} = \delta \leq \frac{1}{2}$ it holds that*

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

▶ **Lemma 8.** *For every $0 < x < \frac{1}{2}$ it holds that*

$$\frac{x}{2\log_2(6/x)} \leq H^{-1}(x) \leq \frac{x}{\log_2(1/x)}.$$

2.1 Coding for interactive communication

2.1.1 Communication protocols

In this section we briefly recall some basic definitions from communication complexity. For more details we refer the reader to [21, 27]. Let $T = (V, E)$ be a complete finite rooted binary tree. Given an internal vertex v in T, define $\mathsf{son}(v, 0)$, $\mathsf{son}(v, 1)$ to be the left son and the right son of v in T, respectively. Extend son for bit strings of length $n \geq 1$ in the natural way and denote by path the function that given $x \in \{0, 1\}^n$, returns the edges on the unique rooted path to $\mathsf{son}(\mathsf{root}(T), x)$. A *communication protocol* π consists of:

- A function $f_v : \{0, 1\}^n \to \{0, 1\}$ for every internal node v in T.
- A label $\mathsf{player}(v) \in \{A, B\}$ for each internal node v.
- A label $\mathsf{value}(v) \in \{0, 1\}$ for every leaf v.

The protocol π induces a function $f = f(\pi) : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ in the following natural way. Given $x, y \in \{0, 1\}^n$, for every internal node $v \in V$, if $\mathsf{player}(v) = A$ let $d = f_v(x)$ and otherwise let $d = f_v(y)$. Let u be the left son of v if $d = 0$ and otherwise let u be the right son of v. Thus, given x, y, from every internal node v goes out exactly one edge $e_v(x, y) = (v, u(x, y))$. Let $E(x, y) = \{e_v(x, y) \mid v \text{ internal node}\}$ be the set of these edges. Observe that the edge set $E(x, y)$ induces a unique root to leaf path in T. Let $v(x, y)$ be that unique leaf that is reachable from the root. We define $f(x, y) = \mathsf{value}(v(x, y))$. We write $\mathsf{depth}(\pi)$ for the depth of T.

The computation above of $f(x, y)$ can be made by two parties, Alice that holds x and Bob that holds y, that can communicate over a channel, in the natural way. Namely, at node v, if $\mathsf{player}(v) = A$ then Alice sends to Bob $f_v(x)$ wheres at a node v with $\mathsf{player}(v) = B$ Bobs sends $f_v(y)$ to Alice. It is clear that the number of bits communicated is the depth of the tree. We say that a protocol is *alternating* if $\mathsf{player}(v) = A$ if and only if v is at even depth. From here on we focus only on alternating protocols.

2.1.2 The pointer jumping game

The pointer jumping game is, in a sense, a complete problem for interactive protocols. Let $T = (V, F)$ be a complete finite rooted binary tree. The depth of a vertex v is the distance, measured in edges, from the root to v. In particular, the depth of the root is 0. We partition the internal nodes of T to $V = V_A \cup V_B$, where V_A contains all nodes at even depth and V_B all nodes at odd depth. We partition the edge set $F = X \cup Y$ with X being the edges going out of V_A and Y the edges leaving V_B. We call a subset of edges $E \subseteq F$ *consistent* if every internal node has exactly one outgoing edge in E. Given a consistent set of edges E, we partition $E = E_A \cup E_B$ where $E_A = E \cap X$ and $E_B = E \cap Y$. It is convenient to represent E_A and E_B by functions $\pi_A : V_A \to \{0,1\}$, $\pi_B : V_B \to \{0,1\}$ as follows: for $v \in V_A$, $\pi_A(v) = 0$ if and only if the edge in E_A that goes out of v is to the left son of v, and similarly for π_B.

Note that in any consistent set of edges E there is a unique root to leaf path. The *pointer jumping game* is a function that given a consistent set of edges E returns the unique leaf reachable from the root using the edge set E. Consider a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and a protocol π for f. Note that for any fixed x, y the task of computing the value $f(x, y)$ is an instance of the pointer jumping game. In that sense, the pointer jumping game is complete. Given a function f as above, it is sometimes convenient to consider a corresponding pointer jumping game of depth $R > n$ in which the edge leaving every vertex of depth larger than n points to its left son (this choice is of course arbitrary and any fixed choice will do).

2.1.3 Resilient protocols and interactive coding schemes

A protocol π is said to be ε-*resilient* if on any pair $x, y \in \{0,1\}^n$, in the above two party computation, both Alice and Bob compute $f(x, y)$ correctly even if at most ε-fraction of the communicated bits are flipped. An *interactive coding scheme* (coding scheme for short) is a function CS_ε, parameterized by $\varepsilon \in [0, 1]$, that gets as input a protocol π and outputs an ε-resilient protocol $\pi_\varepsilon = \mathsf{CS}_\varepsilon(\pi)$ with $f(\pi_\varepsilon) = f(\pi)$. The *rate* of the coding scheme CS_ε is defined by

$$\rho(\mathsf{CS}_\varepsilon) = \inf_\pi \frac{\mathsf{depth}(\pi)}{\mathsf{depth}(\pi_\varepsilon)}.$$

Observe that for the purpose of devising a coding scheme CS_ε one may assume that the inputs x, y are fixed. Thus, it suffices to focus on the problem of devising a coding scheme for the pointer jumping game.

3 Binary Tree Codes: Four Colors Suffice

In this section we prove Theorem 2. We start by setting some notation. Let T be the infinite complete rooted binary tree. We identify length-n paths in T that starts at the root with length-n binary strings in the natural way. Namely, we identify left son and right son with 0 and 1, respectively. Given a node v at depth $n \geq 1$ we define $p_v \in \{0,1\}^n$ to be the string that encodes the (unique) path from the root to v.

An edge-coloring of T by a color set Σ is given by a function, which for ease of readability, we slightly abuse notation and also denote by $T : \{0,1\}^{\mathbb{N}_1} \to \Sigma^{\mathbb{N}_1}$, where the color of an edge $e = u \to v$ is $T(p_v)_{\mathsf{depth}(v)}$. Note that T is an online function, namely, for every $x \in \{0,1\}^{\mathbb{N}_1}$ and $i \in \mathbb{N}_1$, the value $T(x)_i$ is determined by $x_1, \ldots, x_i$.

The (probabilistic) construction

Let $\{R_i\}_{i\in\mathbb{N}_1}$ be a sequence of independent random variables, each is uniformly distributed over $\mathbb{F}_4$–the field of 4 elements. Let $\mathbb{F}_2$ be the (unique) subfield of $\mathbb{F}_4$ of size 2. Define the (random) coloring function $T : \mathbb{F}_2^{\mathbb{N}_1} \to \mathbb{F}_4^{\mathbb{N}_1}$ (where we identify $\mathbb{F}_2$ and $\{0,1\}$ in the natural way) as follows: for every $t \in \mathbb{N}_1$

$$T(x)_t = \sum_{i=1}^{t} R_{t+1-i} x_i. \tag{1}$$

▶ **Definition 9.** *Let v be a depth-n vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. For $k = 1, \ldots, \ell$ we define the random variable*

$$a_v(x, y, k) = T(p_v \circ 1 \circ y)_{n+k} - T(p_v \circ 0 \circ x)_{n+k}.$$

Note that $a_v(x, y, k)$ is a (random) element in $\mathbb{F}_4$. We define the integral random variable

$$h_v(x, y) = \sum_{k=1}^{\ell} I_k,$$

where I_k is the indicator random variable that equals 1 when $a_v(x, y, k) \neq 0$. Note that $h_v(x, y) \in \{0, 1, \ldots, \ell\}$ is the Hamming distance between $T(p_v \circ 0 \circ x)_{[n+1,n+\ell]}$ and $T(p_v \circ 1 \circ y)_{[n+1,n+\ell]}$.

▷ Claim 10. Let v be a vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, for every $k \in \{1, \ldots, \ell\}$ it holds that

$$a_v(x, y, k) = R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i.$$

Proof. Denote the depth of v by n. Fix $k \in \{1, \ldots, \ell\}$. By Equation (1),

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i.$$

Thus,

$$\begin{aligned} a_v(x, y, k) &= \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i - \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i. \end{aligned}$$

◁

▷ Claim 11. Let v be a vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, the random variables $a_v(x, y, 1), \ldots, a_v(x, y, \ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$.

Proof. By Claim 10, $a_v(x,y,k) = R_k + L_k$ where L_k is some $\mathbb{F}_4$-linear combination of $R_1,\ldots,R_{k-1}$. Therefore, $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1)$, $\ldots, a_v(x,y,k-1)$. As this holds for every k we have that $a_v(x,y,1),\ldots,a_v(x,y,\ell)$ are independent. To conclude the proof, note that for every fixing of $R_1,\ldots,R_{k-1}$, $a_v(x,y,k) = R_k + \ell_k$ for some fixed $\ell_k \in \mathbb{F}_4$ and so $a_v(x,y,k)$ is uniform over $\mathbb{F}_4$. ◁

▷ Claim 12. For every two vertices u,v in T and every $x,y \in \mathbb{F}_2^{\ell-1}$,

$$h_v(x,y) = h_u(x,y),$$
$$h_v(x,y) = h_v(0^{\ell-1}, y-x).$$

Proof. The first equality follows immediately by Claim 10 as, for every $k \in \{1,\ldots,\ell\}$, the expression obtained for $a_v(x,y,k)$ is independent of the choice of v. As for the second asserted equality, again by Claim 10,

$$\begin{aligned} a_v(x,y,k) &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y-x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}((y-x)-0)_i \\ &= a_v(0^{\ell-1}, y-x, k), \end{aligned}$$

where observe that for the last equality we are using the fact that $\mathbb{F}_2$ is a subfield of $\mathbb{F}_4$ and so $y-x \in \mathbb{F}_2^{\ell-1}$. Indeed, recall that a_v's second argument is a binary string and so the equality above would have been meaningless otherwise. The above equation implies $h_v(x,y) = h_v(0^{\ell-1}, y-x)$, proving the claim. ◁

Given Claim 12 we can simplify our notation as follows. Let r denote the root of T. For $x \in \{0,1\}^{\ell-1}$ and $k \in \{1,\ldots,\ell\}$ we define the random variables

$$a(x,k) = a_r(0^\ell, 1 \circ x, k),$$
$$h(x) = h_r(0^{\ell-1}, x).$$

Note that $h(x) = \sum_{k=1}^{\ell} a(x,k)$.

▶ **Theorem 13.** *There exists a fixing of the sequence $\{R_i\}_i$ such that the function T is a tree code with distance* 0.05.

Proof. First note that for every fixing of the sequence $\{R_i\}_i$, T is an online function. Observe that, for a fixing of $\{R_i\}_i$, T is a tree code with distance δ if and only if for every $\ell \geq 1$ and $x \in \{0,1\}^{\ell-1}$ it holds that $h(x) \geq \delta\ell$. Indeed, recall that by definition, T is a tree code with distance δ if and only if for every vertex v in T, $\ell \geq 1$, and for every $x,y \in \{0,1\}^{\ell-1}$ it holds that $h_v(x,y) \geq \delta\ell$. However, by Claim 12, $h_v(x,y) = h(y-x)$.

For $x \in \{0,1\}^{\ell-1}$ denote by $E(x)$ the event $h(x) < \delta\ell$. By the above discussion, it suffices to prove, for $\delta = 0.05$, that

$$\Pr\left[\bigcup_{x\in\{0,1\}^{\mathbb{N}}} E(x)\right] < 1.$$

To this end, by the union bound, it suffices to prove that

$$\sum_{x\in\{0,1\}^{\mathbb{N}}} \Pr[E(x)] < 1.$$

Consider any $x \in \{0,1\}^{\ell-1}$ with $\ell \geq 1$. Note that the event $E(x)$ holds if and only if there exists a set $T \subseteq \{1,\ldots,\ell\}$ of size $|T| \geq \lceil (1-\delta)\ell \rceil$ such that for every $k \in T$, $a(x,k) = 0$. By taking the union bound over all such sets T, and using that $a(x,1),\ldots,a(x,\ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$ (Claim 11), we get

$$\Pr[E(x)] \leq \binom{\ell}{\lceil (1-\delta)\ell \rceil} 4^{-\lceil (1-\delta)\ell \rceil}. \tag{2}$$

By Lemma 7, we have that

$$\frac{1}{\ell} \cdot \log_2 \binom{\ell}{\lceil (1-\delta)\ell \rceil} \leq H\left(\frac{\lceil (1-\delta)\ell \rceil}{\ell}\right).$$

As $\delta < \frac{1}{2}$ and since the entropy function H decreases in $[\frac{1}{2}, 1]$ we have that

$$H\left(\frac{\lceil (1-\delta)\ell \rceil}{\ell}\right) \leq H(1-\delta) = H(\delta).$$

Substitute to Equation (2), we get that

$$\Pr[E(x)] \leq 2^{(H(\delta)-2(1-\delta))\ell}.$$

Thus,

$$\begin{aligned}\sum_{x \in \{0,1\}^{\mathbb{N}}} \Pr[E(x)] &\leq \sum_{\ell=1}^{\infty} 2^{\ell-1} \cdot 2^{(H(\delta)-2(1-\delta))\ell} \\ &= \frac{1}{2} \sum_{\ell=1}^{\infty} 2^{(H(\delta)+2\delta-1)\ell}.\end{aligned}$$

One can verify that for $\delta = 0.05$ the above geometric sum is strictly smaller than 1, and the theorem follows. ◀

3.1 Improving the distance

We now show a method for improving the distance. We illustrate it to obtain a bound of 0.136 on the distance, which proves Theorem 2, though we believe that the method can be used to push the bound further. It is fairly easy to show that the distance of a 4-color tree code cannot be larger than 1/2.

▶ **Theorem 14.** *There exists a fixing of the sequence $\{R_i\}_i$ such that the function T is a tree code with distance* 0.136*.*

Proof. For the proof it will be convenient to consider a specific representation of $\mathbb{F}_4$. We make use of the standard construction of $\mathbb{F}_4$ as a quotient of the polynomial ring over $\mathbb{F}_2$ with respect to an ideal generated by a degree 2 irreducible element as follows. Note that $t^2+t+1 \in \mathbb{F}_2[t]$ is irreducible, and so $K = \mathbb{F}_2[t]/\langle t^2+t+1 \rangle$ is a field of 4 elements which we will take as the construction for $\mathbb{F}_4$. Let α be the class of t in K. In this representation, the field $\mathbb{F}_4$ consists of the elements $0, 1, \alpha, \alpha+1$ where $\alpha^2+\alpha+1 = 0$.

Consider the sequence $\{R_i\}_{i \in \mathbb{N}}$ as in the beginning of the section but with the fixings $R_1 = 1$ and $R_2 = \alpha$. Observe that for every $x \in \mathbb{F}_2^{\ell-1}$ with $\ell \geq 2$ it holds that $a(x,1) = 1$ and $a(x,2) = \alpha + x_1$. In particular, $a(x,1), a(x,2)$ are both non-zeros and so $h(x) \geq 2$. Let

$\ell_0 \geq 2$ be an integer parameter to be chosen later on. By the above, we have that for every $x \in \mathbb{F}_2^{\ell-1}$ with $\ell \leq \ell_0$ it holds that

$$\frac{h(x)}{\ell} \geq \frac{2}{\ell_0}. \tag{3}$$

For $x \in \{0,1\}^{\ell-1}$ denote by $E_{1,\alpha}(x)$ the event $h(x) < \delta\ell$ with the $\{R_i\}_{i\in\mathbb{N}}$ as defined above, namely, $R_1 = 1$, $R_2 = \alpha$ and the rest of the random variables $\{R_i \mid i \geq 3\}$ are independent and uniformly distributed over $\mathbb{F}_4$. Once we establish a bound of

$$\Pr\left[\bigcup_{|x|\geq\ell_0} E_{1,\alpha}(x)\right] < 1 \tag{4}$$

for some choice of δ then, combined with Equation (3), we will establish the existence of a tree code with distance at least

$$\min\left(\frac{2}{\ell_0}, \delta\right).$$

Consider any $x \in \{0,1\}^{\ell-1}$ with $\ell \geq \ell_0 + 1$. The event $E_{1,\alpha}(x)$ holds if and only if there exists a set $T \subseteq \{3,\ldots,\ell\}$ of size $|T| \geq \lceil(1-\delta)\ell\rceil$ such that for every $k \in T$, $a(x,k) = 0$. By taking the union bound over all such sets T, and using that $a(x,3),\ldots,a(x,\ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$, we get that

$$\begin{aligned}\Pr[E_{1,\alpha}(x)] &\leq \binom{\ell-2}{\lceil(1-\delta)\ell\rceil} 4^{-\lceil(1-\delta)\ell\rceil} \\ &\leq \binom{\ell}{\lceil(1-\delta)\ell\rceil} 4^{-\lceil(1-\delta)\ell\rceil}\end{aligned}$$

By Lemma 7, we have that

$$\frac{1}{\ell} \cdot \log_2 \binom{\ell}{\lceil(1-\delta)\ell\rceil} \leq H\left(\frac{\lceil(1-\delta)\ell\rceil}{\ell}\right).$$

As we will choose $\delta < \frac{1}{2}$ and the entropy function H decreases in $[\frac{1}{2}, 1]$ we have that

$$H\left(\frac{\lceil(1-\delta)\ell\rceil}{\ell}\right) \leq H(1-\delta) = H(\delta).$$

Thus,

$$\Pr[E_{1,\alpha}(x)] \leq 2^{(H(\delta)-2(1-\delta))\ell}.$$

By substituting the above equation to Equation (4), we get that

$$\sum_{|x|\geq\ell_0} \Pr[E_{1,\alpha}(x)] \leq \sum_{\ell=\ell_0+1}^{\infty} 2^{\ell-1} \cdot 2^{(H(\delta)-2(1-\delta))\ell}.$$

Write $\beta = 2^{H(\delta)+2\delta-1}$. Then, the above is bounded by

$$\frac{1}{2}\sum_{\ell=\ell_0+1}^{\infty} \beta^\ell = \frac{\beta^{\ell_0+1}}{2(1-\beta)}.$$

Consider the real polynomial

$$f_{\ell_0}(x) = x^{\ell_0+1} - 2(1-x).$$

We have that

$$f'_{\ell_0}(x) = (\ell_0+1)x^{\ell_0} + 2$$

Since $\ell_0 \geq 2$, $f'_{\ell_0}(x) > 0$ for all $x \geq 0$. Further, $f_{\ell_0}(0) = -2$ and $f_{\ell_0}(1) = 1$. Thus, $f_{\ell_0}(x)$ has a single root $\beta_{\ell_0} \in [0,1]$ (in fact, β_{ℓ_0} is monotone-increasing as a function of ℓ_0, and $\beta_{\ell_0} \to 1$ as $\ell_0 \to \infty$). For a fixed choice of ℓ_0, by choosing $\beta < \beta_{\ell_0}$ and solving for δ (recall $\beta = 2^{H(\delta)+2\delta-1}$) to obtain δ_{ℓ_0}, we get that there exists a fixing of $\{R_i \,|\, i \geq 3\}$ such that the obtained tree code has distance at least $\min(\delta_{\ell_0}, \frac{2}{\ell_0})$. Thus, the obtained bound is

$$\max_{\ell_0 \geq 2} \min\left(\delta_{\ell_0}, \frac{2}{\ell_0}\right).$$

One can verify that $\ell_0 = 14$ maximizes the above equation to get distance larger than 0.136. ◀

4 Palette-Alternating Tree Codes

In this section we prove Theorem 4. To this end we recall the definition of the (field) trace function $\mathsf{Tr} : \mathbb{F}_4 \to \mathbb{F}_2$ that is given by $\mathsf{Tr}(x) = x + x^2$. Observe that the trace function is an $\mathbb{F}_2$-linear map whose image and kernel are $\mathbb{F}_2$. In particular, if X is uniform over $\mathbb{F}_4$, then $\mathsf{Tr}(X)$ is uniform over $\mathbb{F}_2$.

Let ε be a given parameter and define $b = \lceil 1/\varepsilon \rceil$. Let $\{R_i\}_{i\in\mathbb{N}}$ be a sequence of independent random variables, each is uniformly distributed over $\mathbb{F}_4$ except that R_1 is fixed to $R_1 = 1$. We define a palette-alternating tree code with b palette sets $\Sigma_0, \ldots, \Sigma_{b-1}$ such that $\Sigma_0 = \mathbb{F}_4$ and $\Sigma_i = \mathbb{F}_2$ for $i > 0$. Let $x \in \mathbb{F}_2^{\mathbb{N}}$. For every $k \in \mathbb{N}$, define

$$S_k(x) = \sum_{i=1}^{k} R_{k+1-i} x_i,$$

where addition and multiplication are performed in $\mathbb{F}_4$ and, as usual, $\mathbb{F}_2$ is identified with the unique subfield of two elements in $\mathbb{F}_4$. The coloring function is given by

$$T(x)_k = \begin{cases} S_k(x), & k \equiv_b 0; \\ \mathsf{Tr}(S_k(x)), & \text{otherwise.} \end{cases}$$

▶ **Theorem 15.** *The function T above is a palette-alternating tree code with rate $1-\varepsilon$ and distance $\delta = \Omega(\varepsilon \log^{-1}(1/\varepsilon))$.*

Proof. First, observe that T is indeed an online function with rate larger than $1-\varepsilon$. Further Definition 9 can be carried over to the more general case of palette-alternating tree codes. We turn to prove an analog to Claim 10.

▷ Claim 16. Let v be a depth-n vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, for every $k \in \{1, \ldots, \ell\}$ it holds that

$$a_v(x, y, k) = \begin{cases} R_k + S_{k-1}(y - x), & n + k \equiv_b 0; \\ \mathsf{Tr}(R_k + S_{k-1}(y - x)), & \text{otherwise.} \end{cases}$$

Proof. Fix $k \in \{1, \ldots, \ell\}$. Assume first that $n + k \equiv_b 0$. Then,

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=n+1}^{n+k} R_{n+k+1-i}(0 \circ x)_{i-n} \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i.$$

Thus,

$$\begin{aligned} a_v(x, y, k) &= \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i - \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i \\ &= R_k + S_{k-1}(y - x). \end{aligned}$$

Assume now that $n + k \not\equiv_b 0$. Using that Tr is $\mathbb{F}_2$-linear,

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \mathsf{Tr}\left(\sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i\right) \\ &= \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=n+1}^{n+k} \mathsf{Tr}\left(R_{n+k+1-i}\right)(0 \circ x)_{i-n} \\ &= \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=1}^{k} \mathsf{Tr}(R_{k+1-i})(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=1}^{k} \mathsf{Tr}(R_{k+1-i})(1 \circ y)_i.$$

Thus, again by $\mathbb{F}_2$-linearity of Tr,

$$\begin{aligned} a_v(x, y, k) &= \mathsf{Tr}(R_k) + \sum_{i=1}^{k-1} \mathsf{Tr}(R_{k-i})(y - x)_i \\ &= \mathsf{Tr}(R_k + S_{k-1}(y - x)). \end{aligned}$$

◁

▷ Claim 17. Let v be a depth-n vertex and $x, y \in \mathbb{F}_2^{\ell-1}$ distinct. Then, the random variables $a_v(x, y, 1), \ldots, a_v(x, y, \ell)$ are independent. Moreover, let $k \in [\ell]$. If $n + k \equiv_b 0$ then $a_v(x, y, k)$ is uniformly distributed over $\mathbb{F}_4$ and otherwise it is uniform over $\mathbb{F}_2$.

Proof. By Claim 16, if $n+k \equiv_b 0$ then $a_v(x,y,k) = R_k + L_k$ where L_k is a linear combination of $R_1, \ldots, R_{k-1}$. Thus, in this case, $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1), \ldots, a_v(x,y,k-1)$. Otherwise, namely $n+k \not\equiv_b 0$, we have that $a_v(x,y,k) = \mathsf{Tr}(R_k + L_k) = \mathsf{Tr}(R_k) + \mathsf{Tr}(L_k)$. Since for every fixing of L_k, $a_v(x,y,k)$ is uniform over $\mathbb{F}_2$, we have that $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1), \ldots, a_v(x,y,k-1)$. As this holds for every $k \in [\ell]$ we have that $a_v(x,y,1), \ldots, a_v(x,y,\ell)$ are independent and their marginal distributions are as stated. ◁

▷ Claim 18. Let u, v be two vertices with depth n, m, respectively such that $n \equiv_b m$. Let $x, y \in \mathbb{F}_2^{\ell-1}$. Then,

$$h_v(x,y) = h_u(x,y),$$
$$h_v(x,y) = h_v(0^{\ell-1}, y-x).$$

Proof. Let $C_k = R_k + S_{k-1}(y-x)$. By Claim 16,

$$a_u(x,y,k) = \begin{cases} C_k, & n+k \equiv_b 0; \\ \mathsf{Tr}(C_k), & \text{otherwise.} \end{cases}$$

As C_k is independent of the choice of u and $n \equiv_b m$ we have that $a_u(x,y,k)$ is the same random variable as $a_v(x,y,k)$. Since this holds for every k, we have that $h_v(x,y) = h_u(x,y)$.

We turn to prove the the second asserted equality. Assume first that $k \in [\ell]$ is such that $n + k \equiv_b 0$. By Claim 16,

$$\begin{aligned} a_u(x,y,k) &= R_k + S_{k-1}(y-x) \\ &= R_k + S_{k-1}((y-x) - 0^{\ell-1}) \\ &= a_u(0^{\ell-1}, y-x, k), \end{aligned}$$

where observe that for the last equality we are using the fact that $\mathbb{F}_2$ is a subfield of $\mathbb{F}_4$ and so $y - x \in \mathbb{F}_2^{\ell-1}$. Indeed, recall that a_v's second argument is a binary string and so the equality above would have been meaningless otherwise. The case $n + k \not\equiv_b 0$ follows by a similar argument and using the $\mathbb{F}_2$-linearity of Tr. ◁

Given Claim 18, we can simplify our notation as follows. Let v_0 denote the root of the tree. For $i = 1, \ldots, b-1$ let v_i denote the left son of v_{i-1}. For every $i \in \{0, 1, \ldots, b-1\}$ and $x \in \{0,1\}^{\ell-1}$ we define the random variables

$$a_i(x,k) = a_{v_i}(0^\ell, 1 \circ x, k),$$
$$h_i(x) = h_{v_i}(0^{\ell-1}, x).$$

Define

$$\delta = c_1 \varepsilon \log^{-1}(1/\varepsilon),$$
$$\ell_0 = 12\lceil \varepsilon^{-1} \log(1/\varepsilon) \rceil,$$

for some constant $c_1 \in [0,1]$ to be set later on. Observe that for every fixing of the sequence $\{R_i\}$, T is a palette-alternating tree code with distance δ if and only if for every $x \in \{0,1\}^{\ell-1}$ and $i \in \{0, 1, \ldots, b-1\}$ it holds that $h_i(x) \geq \delta\ell$. Indeed, by definition, T is a palette-alternating tree code with distance δ if and only if for every vertex v, $\ell \geq 1$, and every distinct $x, y \in \{0,1\}^{\ell-1}$ it holds that $h_v(x,y) \geq \delta\ell$. However, by Claim 18, the random variable $h_v(x,y)$ is the same as the random variable $h_i(y-x)$ for $i = \mathsf{depth}(v) \bmod b$.

For $x \in \{0,1\}^{\ell-1}$ and $i \in \{0,1,\ldots,b-1\}$ denote by $E_i(x)$ the event $h_i(x) < \delta\ell$. Note that as $R_1 = 1$ and since $\mathsf{Tr}(1) = 1$ we have that $h_i(x) \geq 1$ for every x. Thus, for $|x| < \ell_0$ we have that

$$\frac{h(x)}{|x|+1} \geq \frac{1}{\ell_0}.$$

Therefore, in order to prove Theorem 15 it suffices to prove that

$$\Pr\left[\bigcup_{|x|\geq\ell_0}\bigcup_{i=0}^{b-1} E_i(x)\right] < 1.$$

Indeed, this will give a bound of $\min\left(\frac{1}{\ell_0}, \delta\right) = \Omega(\varepsilon \log^{-1}(1/\varepsilon))$ on the distance.

Fix $x \in \{0,1\}^{\ell-1}$ and $i \in \{0,1,\ldots,b-1\}$. Observe that $E_i(x)$ holds if and only if there exists a set $T \subseteq [\ell]$ of size $\lceil(1-\delta)\ell\rceil$ such that for every $k \in T$, $a_i(x,k) = 0$. For ease of readability we ignore the ceiling in the calculations below. Recall that $a_i(x,1),\ldots,a_i(x,\ell)$ are independent. Further, $1 - \frac{1}{b}$ fraction of them are uniform over $\mathbb{F}_2$ whereas the remaining $\frac{1}{b}$ fraction are uniform over $\mathbb{F}_4$. Note that by our choice of parameters, $\delta < 1/b$. Thus, for any $\gamma \geq 0$ and a fixed T, we have that

$$\begin{aligned}\Pr\left[\forall k \in T\ a_i(x,k) = 0\right] &\leq 2^{-\left(1-\frac{1}{b}-\gamma\right)\ell} 4^{-\left(\frac{1}{b}-\delta+\gamma\right)\ell} \\ &\leq 2^{-\left(1-\frac{1}{b}\right)\ell} 4^{-\left(\frac{1}{b}-\delta\right)\ell} \\ &= 2^{-\left(1+\frac{1}{b}-2\delta\right)\ell}.\end{aligned}$$

By taking the union bound over the choice of T, and using Lemma 7, we get that

$$\begin{aligned}\Pr[E_i(x)] &\leq \binom{\ell}{\lceil(1-\delta)\ell\rceil} 2^{-\left(1+\frac{1}{b}-2\delta\right)\ell} \\ &\leq 2^{-\left(1+\frac{1}{b}-2\delta-H(\delta)\right)\ell}.\end{aligned}$$

By the union bound,

$$\begin{aligned}\Pr\left[\bigcup_{|x|\geq\ell_0}\bigcup_{i=0}^{b-1} E_i(x)\right] &\leq \sum_{|x|\geq\ell_0}\sum_{i=0}^{b-1}\Pr[E_i(x)] \qquad (5)\\ &\leq b\cdot\sum_{\ell=\ell_0}^{\infty} 2^{\ell-1}\cdot 2^{-\left(1+\frac{1}{b}-2\delta-H(\delta)\right)\ell} \\ &= \frac{b}{2}\cdot\sum_{\ell=\ell_0}^{\infty} 2^{\left(H(\delta)+2\delta-\frac{1}{b}\right)\ell}.\end{aligned}$$

By taking c_1 sufficiently small and using Lemma 8, we get that $H(\delta) + 2\delta - 1/b \leq -\varepsilon/3$. Therefore, Equation (5) is bounded above by

$$\begin{aligned}b\cdot\sum_{\ell=\ell_0}^{\infty} 2^{-\varepsilon\ell/3} &= b\cdot\frac{2^{-\varepsilon\ell_0/3}}{1-2^{-\varepsilon/3}} \\ &\leq \frac{b\varepsilon^4}{1-2^{-\varepsilon/3}} \\ &\leq \frac{2\varepsilon^3}{1-2^{-\varepsilon/3}},\end{aligned}$$

where the penultimate inequality follows by our choice of ℓ_0 and the last inequality follows since $b = \lceil 1/\varepsilon\rceil$. One can verify that the above is strictly bounded by 1 for any $\varepsilon < 1/3$. ◀

5 The Interactive Coding Scheme

In this section we prove Theorem 6. In the first section we set up the framework over which our coding scheme will be defined. In Section 5.2 we present our coding scheme and Sections 5.3, 5.4 contain the analysis.

5.1 Setting up the framework

Round types

Throughout the scheme Alice and Bob send information in an alternating manner. More precisely, at even rounds Alice would decide on a bit to be sent and at odd rounds, Bob will decide what bit to send. Let $t \geq 0$. If t is even we say that it is an *Alice's round* and otherwise it is a *Bob's round*.

Epochs

We further partition the rounds as follows. Let c be a parameter to be set later on. The protocol is divided to *epochs* where each epoch consists of $2c+2$ rounds. The first epoch starts from round 0 to round $2c+1$ and is denoted by $e_0 = [0, 2c+2)$. The second epoch is denoted by $e_1 = [2c+2, 4c+4)$ and, generally, the k'th epoch consists of rounds $[k(2c+2), (k+1)(2c+2))$. Let t be an Alice's round and consider $m = t \bmod (2c+2)$. If $m = 2c$, then t is referred to as *Alice's bit sync round*, and otherwise, t is called an *Alice's edge round*. Similarly, for t a Bob's round, let $m = t \bmod (2c+2)$. If $m = 2c+1$, then t is a *Bob's bit sync round*. Otherwise, t is called *Bob's edge round*.

We denote by $\mathsf{edges}(e)$ the sequence of $2c$ bits sent throughout the edge rounds during epoch e, and define $\mathsf{sync}_A(e), \mathsf{sync}_B(e)$ the bits sent by Alice and Bob during their sync rounds, respectively.

Rewinding mechanism

As the adversary introduce some fraction of errors, the coding scheme should incorporate a "regret mechanism" using which the parties can revert back parts of the already exchanged messages. To formalize that, we will make use of the pair of functions

$$\mathsf{rewind} : \{S, X\}^* \to \{S, X\}^*,$$
$$\mathsf{survive} : \{S, R, X\}^* \to \{S, X\}^*,$$

which are defined as follows. Let $n \geq 1$. We define $\mathsf{rewind}(X^n) = X^n$. Let $v \in \{S, X\}^n \setminus \{X^n\}$ and denote $i \in [n]$ the largest index such that $v_i = S$. Then,

$$\mathsf{rewind}(v)_j = \begin{cases} v_j & j \neq i; \\ X & j = i. \end{cases}$$

We define the function $\mathsf{survive}$ recursively as follows. Let $v \in \{S, R, X\}^n$,

$$\mathsf{survive}(v) = \begin{cases} \mathsf{rewind}(\mathsf{survive}(v_0, \ldots, v_{n-1})) \circ X & v_n = R; \\ \mathsf{survive}(v_0, \ldots, v_{n-1}) \circ v_n & v_n \neq R. \end{cases}$$

Decoding the pointer jumping path

We describe now how to decode a rooted path in the pointer jumping game from the bits that were sent during a sequence of epochs. To formalize that, we define the function PJPath that given a sequence of epochs $(e_0, \ldots, e_n)$, computes a rooted path in the depth-n tree T as follows. Define $h : e \to \{S, R\}$ by

$$h(e) = R \quad \iff \quad \mathsf{sync}_A(e) \vee \mathsf{sync}_B(e) = 1$$

where an epoch is initialized with $\mathsf{sync}_A(e) = \mathsf{sync}_B(e) = 0$. Denote $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and let $i_1 < \cdots < i_\ell$ be the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. Finally, set

$$\mathsf{PJPath}(e_0, \ldots, e_n) = \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})).$$

where path is defined in the preliminaries.

▷ Claim 19. Let $e_0, \ldots, e_{n+1}$ be a sequence of epochs such that $\mathsf{sync}_A(e_{n+1}) \vee \mathsf{sync}_B(e_{n+1}) = 0$, then

$$v(\mathsf{PJPath}(e_0, \ldots, e_n)) = \mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_{n+1})), 2c).$$

If on the other hand $\mathsf{sync}_A(e_{n+1}) \vee \mathsf{sync}_B(e_{n+1}) = 1$, then

$$\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1})).$$

Proof. For the first direction of the claim, note that as $h(e_{n+1}) = S$ it follows that

$$\mathsf{survive}(h(e_0), \ldots, h(e_{n+1})) = \mathsf{survive}(h(e_0), \ldots, h(e_n)) \circ S.$$

Let $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and $0 \leq i_1 < \cdots < i_\ell \leq n$ where $\ell \geq 0$, the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. Thus, the set of indices that corresponds to an S symbol in $\mathsf{survive}(h(e_0), \ldots, h(e_n)) \circ S$ is exactly $\{i_1, \ldots, i_\ell, n+1\}$. Hence,

$$\begin{aligned} \mathsf{PJPath}(e_0, \ldots, e_n) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})); \\ \mathsf{PJPath}(e_0, \ldots, e_{n+1}) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell}) \circ \mathsf{edges}(e_{n+1})), \end{aligned}$$

and so $\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1}))$.

For the other direction, by definition, $h(e_{n+1}) = R$ and so

$$\mathsf{survive}(h(e_0), \ldots, h(e_{n+1})) = \mathsf{rewind}(\mathsf{survive}(h(e_0), \ldots, h(e_n))) \circ X.$$

Let $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and $i_1 < \cdots < i_\ell$ the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. By the definition of the rewind function, if $\ell > 0$ then the indices $i_1, \ldots, i_{\ell-1}$ correspond to an S symbol in $\mathsf{rewind}(\mathsf{survive}(h(e_0), \ldots, h(e_n)))$. Thus,

$$\begin{aligned} \mathsf{PJPath}(e_0, \ldots, e_n) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})); \\ \mathsf{PJPath}(e_0, \ldots, e_{n+1}) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_{\ell-1}})). \end{aligned}$$

Therefore, $\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1}))$ as stated. In the case that $\ell = 0$, recall that $\mathsf{root}(T) = \mathsf{ancestor}(\mathsf{root}(T), m)$ for all $m \in \mathbb{N}$ concluding the proof. ◁

Transcript notations

Let $\mathcal{TC}$ be the palette-alternating tree code from Theorem 15 set with distance parameter $\delta_{\mathcal{TC}}$ whose value will be set later on. Denote by $\mathsf{TCEnc}, \mathsf{TCDec}$ the encoding and decoding functions of $\mathcal{TC}$, respectively where we decode to minimize the distance from the received word to a codeword. At every round, one of the parties would decide on a bit to be sent. That bit is not sent over the channel as is but rather is encoded using a palette-alternating tree code. For an even integer $t \geq 0$ we denote by $(a_0, a_2, \ldots, a_t)$ those bits that Alice would "like" to send from round 0 until round t. As mentioned above, the actual symbols that Alice sends are obtained by encoding these bits using $\mathcal{TC}$. Similarly, for an odd $t \geq 1$ we denote $(b_1, b_3, \ldots, b_t)$ the bits Bob would like to send. For an even integer $t \geq 0$ we define $\tilde{a}(t) = (\tilde{a}(t)_0, \tilde{a}(t)_2, \ldots, \tilde{a}(t)_t)$ to be the bits that are decoded, via TCDec, given the received transmission to Bob at round t. Note that $\tilde{a}(t)_i$ may not equal $\tilde{a}(t')_i$ for distinct times t, t', and certainly may not equal a_i.

For an odd t, we define $r_A(t) = (a_0, \tilde{b}(t)_1, a_2, \tilde{b}(t)_3, \ldots, \tilde{b}(t)_t)$ and similarly for an even t, $r_B(t) = (\tilde{a}(t)_0, b_1, \tilde{a}(t)_2, b_3, \ldots, \tilde{a}(t)_t)$. We further define $r(t) = (a_0, b_1, a_2, \ldots, b_t)$ for odd t and $r(t) = (a_0, b_1, a_2, \ldots, a_t)$ for even t. Recall that for a given set of edges E', we defined $v(E')$ to be the unique vertex in T with largest depth that is reachable from the root using the edge set E'. We define

$$p_A(t) = \mathsf{PJPath}(r_A(t));$$
$$\gamma_A(t) = v(p_A(t));$$
$$\alpha_A(t) = v(p_A(t) \cap (E_A \cup Y)).$$

Similarly,

$$p_B(t) = \mathsf{PJPath}(r_B(t));$$
$$\gamma_B(t) = v(p_B(t));$$
$$\alpha_B(t) = v(p_B(t) \cap (E_B \cup X)).$$

5.2 The coding scheme

The coding scheme is composed of two parts. The first consists of R rounds and the second of additional $2\tau R$ rounds where τ is a parameter to be chosen later on. We turn to describe the first part of the scheme. The second part is described in Section 5.2.2.

5.2.1 Part 1 of the coding scheme

We present the scheme from Alice's point of view. The scheme from Bob's point of view can be easily inferred. As mentioned, Alice's algorithm is partitioned to epochs. At the first round of epoch $e_k = [k(2c+2), (k+1)(2c+2))$ Alice computes $v_A = \gamma_A(k(2c+2)-1)$. We will make sure to maintain the invariant that at odd times t, $\gamma_A(t) \in V_A$. In particular, $v_A \in V_A$. For each round type, Alice proceeds as follows:

5.2.1.1 Alice's edge round

Let t be an Alice's edge round, namely, t is an even integer with $t \not\equiv 2c \pmod{2c+2}$.

1. At the edge rounds, Alice maintains v_A in order to choose a_t which is the bit that she would like to send at round t. Alice sets $a_t \leftarrow \pi_A(v_A)$. This operation is well-defined as we will be making sure also to maintain the invariant that in Alice's edge rounds $v_A \in V_A$.
2. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$.
3. Update $v_A \leftarrow \mathsf{son}(v_A, a_t)$.

5.2.1.2 Bob's edge round

Let t be Bob's edge round, namely, t is odd with $t \not\equiv 2c+1 \pmod{2c+2}$. At this round Bob sent a bit to Alice who, in turn, proceeds by updating v_A as follows:

1. $v_A \leftarrow \mathsf{son}(v_A, \tilde{b}(t)_t)$, where, recall $\tilde{b}(t)$ is the bit-string Alice decoded from the received transcript at round t.

5.2.1.3 Alice's bit sync round

Let $t \equiv 2c \pmod{2c+2}$. Notice that $\alpha_A(t-1)$ is an ancestor of $\gamma_A(t-1)$. We consider the following cases according to $\alpha_A(t-1)$, $\gamma_A(t-1)$ locations:

1. If $\alpha_A(t-1) = \gamma_A(t-1)$, then
   a. $a_t \leftarrow 0$ (0 encodes "hold")
   b. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$
2. If $\alpha_A(t-1)$ is a strict ancestor of $\gamma_A(t-1)$ then
   a. $a_t \leftarrow 1$ (1 encodes "revert")
   b. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$

5.2.2 Part 2 of the coding scheme

Recall that the coding scheme is divided to two parts. We now present the second part which take place during rounds $[R, (1+2\tau)R]$. This part is not partitioned to epochs and we describe it per round. We define the function $\mathsf{counter}_A : V \to \mathbb{N}$ that is initialized to 0. Recall that n denotes the depth of the tree T. More precisely, our convention is that edges leaving vertices of depth larger than n always point to their left son.

5.2.2.1 Alice's edge round

Let t be an Alice's round, namely, t is an even integer.

1. Alice sets $a_t \leftarrow 0$.
2. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$.

5.2.2.2 Bob's edge round

Let t be a Bob's round, namely, t is odd. At this round, Bob sent a bit to Alice who, in turn, proceeds by updating $\mathsf{counter}_A$ as follows:

1. Alice computes $\gamma_A(t)$.
2. If $\mathsf{depth}(\gamma_A(t)) \geq n$, denote by v the unique ancestor of $\gamma_A(t)$ of depth n. Alice sets $\mathsf{counter}_A(v) = \mathsf{counter}_A(v) + 1$.

5.2.2.3 Final round

Alice returns the vertex v that maximizes $\mathsf{counter}_A(v)$. The analysis will show that such vertex exists and is unique.

5.2.2.4 Remark

Note that in most rounds, TCEnc outputs a symbol in $\mathbb{F}_2$ which corresponds to a single bit transmitted. At the rounds in which the symbol is an $\mathbb{F}_4$-element, we send the information in two rounds and the round of the other party in between is ignored. For simplicity, we make this issue transparent to the coding scheme.

5.3 A simpler analysis with sub-optimal rate

In this section we prove that the coding scheme above, when set with suitable parameters $\delta_{\mathcal{TC}}$, c, τ, has rate $1-\tilde{O}(\sqrt[3]{\varepsilon})$. Many of the ideas and results used in this section will be used for the proof of Theorem 6, to be presented in Section 5.4, which requires additional ideas. We assume R is an integral multiple of $2c+2$ and let k be the number of epochs, namely, $R=(2c+2)k$.

Good rounds

We say that $t \in [R]$ is *good* if the decoding at round t succeeds. More precisely, when t is even, round t is good if

$$(a_0, a_2, \ldots, a_t) = (\tilde{a}(t)_0, \tilde{a}(t)_2, \ldots, \tilde{a}(t)_t).$$

Similarly, an odd t is good if

$$(b_1, b_3, \ldots, b_t) = (\tilde{b}(t)_1, \tilde{b}(t)_3, \ldots, \tilde{b}(t)_t).$$

We make use of the following lemma proved by Schulman [29] (see also Section 2.1.3 in [12]).

▶ **Lemma 20** ([29]). *Let $\mathcal{TC}$ be a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$. Assume the channel has at most ε-fraction errors. Then, at most*

$$\mu \triangleq 2\varepsilon/\delta_{\mathcal{TC}}$$

fraction of rounds are bad.

Good epochs

We say that epoch $e=[t, t+2c+2)$ is *good* if each round $r \in [t-1, t+2c]$ is good and otherwise we call it *bad*. Note that for an epoch to be good we require that the last round of the previous epoch is good though do not require the last round of the current epoch to be good. Note further that at least $1-(2c+2)\mu$ fraction of the epochs are good. We wish to define vertices analog to $\gamma_A(t), \alpha_A(t)$ and $\gamma_B(t), \beta_B(t)$ that are defined according to what was actually sent by the parties in the first t rounds rather than according to what was received. Formally, define

$$\gamma(t) = v(\mathsf{PJPath}(r(t)));$$
$$\alpha(t) = v(\mathsf{PJPath}(r(t)) \cap (E_A \cup Y));$$
$$\beta(t) = v(\mathsf{PJPath}(r(t)) \cap (E_B \cup X)),$$

where recall that $r(t)$ is defined in the paragraph presenting our transcript notations in Section 5.1. Let $v(t)$ be the least common ancestor of $\alpha(t), \beta(t)$ in T. Observe that $v(t)$ is equal to either $\alpha(t)$ or $\beta(t)$ and in particular is an ancestor of $\gamma(t)$.

▷ Claim 21. Let $e=[t, t+2c+2)$ be a good epoch such that $v(t-1) \neq \gamma(t-1)$. Then,

$$\mathsf{sync}_A(e) = 1 \vee \mathsf{sync}_B(e) = 1.$$

Proof. Observe that the hypothesis of the claim implies that $v(t-1)$ is a strict ancestor of $\gamma(t-1)$. As $\gamma(t-1)$ is a strict ancestor of $\gamma(t+2c-1)$ and since $v(t+2c-1) = v(t-1)$ it follows that $v(t+2c-1) \neq \gamma(t+2c-1)$. As round $t+2c-1$ is good, it holds that

$$\gamma_A(t+2c-1) = \gamma(t+2c-1) = \gamma_B(t+2c-1).$$

Furthermore, by the definition of α_A and β_B it follows that

$$\alpha_A(t+2c-1) = \alpha(t+2c-1);$$
$$\beta_B(t+2c-1) = \beta(t+2c-1).$$

Thus, as $v(t+2c-1) \neq \gamma(t+2c-1)$, at least one of the following holds $\alpha_A(t+2c-1) \neq \gamma_A(t+2c-1)$ or $\beta_B(t+2c-1) \neq \gamma_B(t+2c-1)$. Hence, at least one of the parties set its sync bit to 1. ◁

Short-split epochs

We define the indicator function

$$\mathsf{nearAncestor}(v(t), \gamma(t)) = \begin{cases} 1 & \mathsf{dist}(v(t), \gamma(t)) \in (0, 2c); \\ 0 & \text{otherwise.} \end{cases}$$

A good epoch $e = [t, t+2c+2)$ is called a *short-split* epoch if

$$\mathsf{nearAncestor}(v(t-1), \gamma(t-1)) = 1.$$

▷ Claim 22. The number of short-split epochs is bounded above by the number of bad epochs.

Proof. Consider any two short-split epochs $e = [t, t+2c+2), e' = [t', t'+2c+2)$ with $t < t'$. Since e is a short-split epoch, then e is good and also $v(t-1) \neq \gamma(t-1)$. By Claim 21, Alice or Bob set their sync bit to 1. By Claim 19 it holds that $\gamma(t+2c+1) = \mathsf{ancestor}(\gamma(t-1), 2c)$. Observe that as $d < 2c$, this results in $v(t+2c+1) = \gamma(t+2c+1)$.

Observe further that, until the arrival of a bad epoch, at epoch $e'' = [t'', t''+2c+2)$ we have that $v(t''-1) = \gamma(t''-1)$. Since e' is a short-split epoch, $v(t'-1) \neq \gamma(t'-1)$. It then follows that there exists a bad epoch preceding e'. Since the first epoch is not short-split, the claim follows. ◁

Potential function for the progress

For an integer $i \geq 0$ and $t = (2c+2)i - 1$, consider the following potential function

$$\Phi(t) = 2\mathsf{depth}(v(t)) - \mathsf{depth}(\gamma(t)).$$

Recall that $\mathsf{depth}(\gamma(t)) \geq \mathsf{depth}(v(t))$ and so when $\Phi(t) \geq n$ it holds that $\mathsf{depth}(v(t)) \geq n$.

▷ Claim 23. If $e = [t, t+2c+2)$ is a good epoch that is not short-split, then $\Phi(t+2c+1) = \Phi(t-1) + 2c$. Otherwise, $\Phi(t+2c+1) \geq \Phi(t-1) - 6c$.

Proof. By Claim 19, $\mathsf{dist}(\gamma(t+2c+1), \gamma(t-1)) \leq 2c$. Observe that by Claim 19 and by the definition of v it follows that $\mathsf{dist}(v(t+2c+1), v(t-1)) \leq 2c$ as well. Thus, the assertion $\Phi(t+2c+1) \geq \Phi(t-1) - 6c$ follows. Let then e be a good epoch that is not short-split, and consider the following cases:

1. First assume that $v(t-1) = \gamma(t-1)$. As epoch e is good, it follows that at the edge rounds, Alice and bob extends the same (correct) path, and so

$$v(t+2c-1) = \gamma(t+2c-1). \tag{6}$$

Since round $t+2c-1$ is good,

$$\gamma_A(t+2c-1) = \gamma(t+2c-1) = \gamma_B(t+2c-1).$$

The above equation together with Equation (6) implies that

$$\alpha_A(t+2c-1) = \gamma_A(t+2c-1);$$
$$\beta_B(t+2c-1) = \gamma_B(t+2c-1).$$

By the algorithm both Alice and Bob sets their sync bit to 0, namely, $\mathsf{sync}_A(e) = \mathsf{sync}_B(e) = 0$. Thus, together with Claim 19 and Equation (6),

$$\mathsf{depth}(\gamma(t+2c+1)) = \mathsf{depth}(\gamma(t-1)) + 2c,$$
$$\mathsf{depth}(v(t+2c+1)) = \mathsf{depth}(v(t-1)) + 2c,$$

and it follows that $\Phi(t+2c+1) = \Phi(t-1) + 2c$.

2. Consider now the case that $v(t-1)$ is a strict ancestor of $\gamma(t-1)$. By Claim 21 it follows that $\mathsf{sync}_A(e) = 1$ or $\mathsf{sync}_B(e) = 1$. Then, by Claim 19 it holds that $\mathsf{ancestor}(\gamma(t-1), 2c) = \gamma(t+2c+1)$. Since e is not a short-split epoch, $v(t-1)$ is an ancestor of $\gamma(t+2c+1)$, and by the definition of v this implies $v(t+2c+1) = v(t-1)$. Thus, it holds that

$$\mathsf{depth}(\gamma(t+2c+1)) = \mathsf{depth}(\gamma(t-1)) - 2c,$$
$$\mathsf{depth}(v(t+2c+1)) = \mathsf{depth}(v(t-1)),$$

and

$$\Phi(t+2c+1) = \Phi(t-1) + 2c,$$

concluding the proof. ◁

By Claim 22, there are at least $(1-2(2c+2)\mu)k$ good epochs which are not short-split. By Claim 23, Φ increases by at least $2c$ in every such epoch. In the remaining epochs, Φ decreases by at most $6c$. Since $\Phi(-1) = 0$ we have that

$$\begin{aligned}\Phi(R) &\geq ((1-2(2c+2)\mu)2c + 2(2c+2)\mu \cdot (-6c))k \\ &= (1-8(2c+2)\mu) \cdot 2ck \\ &= \left(1 - \left(\frac{4}{2c+2} + 16c\mu\right)\right) R.\end{aligned}$$

By setting c to be an integer $c = \Theta(1/\sqrt{\mu})$, we get $\Phi(R) = \left(1 - \Theta(\sqrt{\mu})\right) R$. Now setting $R = (1+\Theta(\sqrt{\mu}))n$, the first part of the scheme assures that $\mathsf{depth}(v(R)) \geq n$.

Analysis of part 2 of the scheme

Let v_{pj} be the unique ancestor of $v(R)$ of depth n in T, it is well defined as the analysis of Part 1 of the scheme assures that $\mathsf{depth}(v(R)) \geq n$. Recall that the second part of the scheme contains $2\tau R$ rounds. By Lemma 20, there are at most

$$(1+2\tau)\mu R$$

bad rounds. By the algorithm, for every good odd round in Part 2 of the scheme, Alice increases $\mathsf{counter}_A(v_{\mathrm{pj}})$ by 1. Observe that in the final round Alice returns the vertex that maximizes $\mathsf{counter}_A$ and so the assertion $\mathsf{counter}_A(v_{\mathrm{pj}}) > (\tau R)/2$ implies that the simulation will terminates successfully. By setting $\tau = 6\mu$, we get

$$\frac{\tau}{2}R > (1+2\tau)\mu R$$

which guarantees that the majority of both Alice's and Bob's rounds in the second part of the coding scheme are good. This concludes the proof of the theorem.

Calculating the rate

At each round of the simulation, a palette-alternating tree code symbol is sent instead of a single bit. By Theorem 4 $\mathcal{TC}$ has rate $1 - O(\delta_{\mathcal{TC}} \log(1/\delta_{\mathcal{TC}}))$. Setting $\delta_{\mathcal{TC}} = \sqrt[3]{\varepsilon}$, we get that the simulation uses

$$\left(1 + O\left(\sqrt{\frac{\varepsilon}{\delta_{\mathcal{TC}}}}\right)\right)\left(1 + O\left(\delta_{\mathcal{TC}} \log\left(\frac{1}{\delta_{\mathcal{TC}}}\right)\right)\right) n = \left(1 + O\left(\sqrt[3]{\varepsilon} \log\left(\frac{1}{\varepsilon}\right)\right)\right) n$$

bits. Thus, the coding scheme rate is $1 - \tilde{O}(\sqrt[3]{\varepsilon})$ as stated.

5.4 Optimal analysis

In this section we prove Theorem 6. We make use of the same coding scheme analyzed in Section 5.3. The improved analysis follows by applying a more delicate analysis of the bad rounds locations as a function of the errors introduced by the adversary.

Let $\mathcal{TC}$ be a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$. Denote by $\mathcal{E} = \{e_1, \ldots, e_{\varepsilon R}\}$ the set of rounds at which the adversary has introduced errors, where $0 \le e_1 < \cdots < e_{\varepsilon R} \le R$. A set of consecutive errors $C = \{e_j, \ldots, e_{j+r-1}\}$ is called a *cluster of errors* (with respect to $\mathcal{TC}$ or more precisely $\delta_{\mathcal{TC}}$) if

$$\forall \ell \in [r-1] \quad e_{j+\ell} - e_j \le \frac{2\ell}{\delta_{\mathcal{TC}}}\,.$$

We define the *cluster interval* of C by $\mathcal{I}(C) = [e_j, e_j + 2r/\delta_{\mathcal{TC}}]$. We denote by $\mathcal{C}$ the set of all clusters (with respect to $\mathcal{E}$).

▷ Claim 24. Let $C_1, C_2 \in \mathcal{C}$ with $C_1 \subseteq C_2$. Then, $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$.

Proof. Let $C_1 = \{e_i, \ldots, e_j\}, C_2 = \{e_m, \ldots, e_k\}$ with $m \le i \le j \le k$. By definition, it holds that $\mathcal{I}(C_1) = [e_i, e_i + 2(j-i+1)/\delta_{\mathcal{TC}}]\,, \mathcal{I}(C_2) = [e_m, e_m + 2(k-m+1)/\delta_{\mathcal{TC}}]$. As $e_i \in C_2$ we have that $e_i \le e_m + 2(i-m)/\delta_{\mathcal{TC}}$, and so

$$\begin{aligned} e_i + \frac{2(j-i+1)}{\delta_{\mathcal{TC}}} &\le e_m + \frac{2(j-m+1)}{\delta_{\mathcal{TC}}} \\ &\le e_m + \frac{2(k-m+1)}{\delta_{\mathcal{TC}}}\,, \end{aligned}$$

which, together with $e_m \le e_i$, concludes the proof. ◁

We will be interested to study clusters on sub-intervals of $[0, R]$ and in particular we wish to consider clusters that are, in a sense, maximal in the sub-interval. To formalize that, let $[a, b]$ be a sub-interval of $[0, R]$. A cluster $C \in \mathcal{C}$ with $C \subseteq [a, b]$ is called $[a, b]$-*maximal* if for

every cluster $C' \subseteq [a,b]$ such that $C \subseteq C'$ it holds that $C' = C$. A $[0,R]$-maximal cluster is simply called maximal. We denote by $\mathcal{M}_{[a,b]}$ the set of all $[a,b]$-maximal clusters, and by $\mathcal{M}$ the set of all maximal clusters.

▷ Claim 25. Every $C_1, C_2 \in \mathcal{M}_{[a,b]}$ are either equal or disjoint.

The proof of the above claim is straightforward. Indeed, by adapting the proof of Claim 24, if false $C_1 \cup C_2 \in \mathcal{C}$ in contradiction to the maximality.

▷ Claim 26. Let $C_1, C_2 \in \mathcal{M}_{[a,b]}$ distinct. Then, $\mathcal{I}(C_1) \cap \mathcal{I}(C_2) = \emptyset$.

Proof. By Claim 25 we have that $C_1 \cap C_2 = \emptyset$, and so we may denote $C_1 = \{e_i, \ldots, e_{i+j}\}$, $C_2 = \{e_m, \ldots, e_{m+n}\}$ with $i+j < m$. Assume toward a contradiction that $\mathcal{I}(C_1) \cap \mathcal{I}(C_2) \neq \emptyset$, and so $e_m \in [e_i, e_i + 2(j+1)/\delta_{\mathcal{TC}})$. Observe that this would imply that $C' = \{e_i, \ldots, e_m\} \in \mathcal{C}$, which together with $C' \subseteq [a,b]$, stands in contradiction to $C_1 \in \mathcal{M}_{[a,b]}$. ◁

▷ Claim 27.

$$\left| \bigcup_{M \in \mathcal{M}} \mathcal{I}(M) \right| \leq \frac{2\varepsilon R}{\delta_{\mathcal{TC}}} .$$

Proof. By Claim 26, and since $|\mathcal{I}(C)| = 2|C|/\delta_{\mathcal{TC}}$ for every $C \in \mathcal{C}$,

$$\left| \bigcup_{M \in \mathcal{M}} \mathcal{I}(M) \right| = \sum_{M \in \mathcal{M}} \frac{2|M|}{\delta_{\mathcal{TC}}} .$$

As all maximal clusters are disjoint (Claim 25),

$$\sum_{M \in \mathcal{M}} |M| \leq \varepsilon R,$$

which concludes the proof. ◁

▶ **Lemma 28.** *Let $r \in [0,R]$. If $r \notin \bigcup_{C \in \mathcal{C}} \mathcal{I}(C)$ then r is a good round.*

Proof. Denote by σ_t the palette-alternating tree code symbol that is sent at round t, and let $\tilde{\sigma}_t$ be the received symbol at that round. Denote by $(\mu_1, \ldots, \mu_r)$ the path on $\mathcal{TC}$ that corresponds to the decoded codeword . Assume toward a contradiction that r is bad, namely, $(\sigma_1, \ldots, \sigma_r) \neq (\mu_1, \ldots, \mu_r)$. Let $\ell \in [r]$ be the largest integer such that $\mu_{r-\ell} \neq \sigma_{r-\ell}$. As $\mathsf{TCDec}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_r)$ returns the codeword that minimizes the distance, and since $\mu_i = \sigma_i$ for every $i < r - \ell$, we have that

$$\Delta((\mu_{r-\ell}, \ldots, \mu_r), (\tilde{\sigma}_{r-\ell}, \ldots, \tilde{\sigma}_r)) \leq \Delta((\tilde{\sigma}_{r-\ell}, \ldots, \tilde{\sigma}_r), (\sigma_{r-\ell}, \ldots, \sigma_r)). \tag{7}$$

Since $\mathcal{TC}$ is a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$,

$$\Delta((\mu_{r-\ell}, \ldots, \mu_r), (\sigma_{r-\ell}, \ldots, \sigma_r)) \geq (\ell+1)\delta_{\mathcal{TC}} . \tag{8}$$

Let $I = \mathcal{E} \cap [r-\ell, r]$, i.e the set of all rounds i such that $\sigma_i \neq \tilde{\sigma}_i$ in the interval $[r-\ell, r]$. Denote $|I| = k$. As $\mathcal{M}_{[r-\ell,r]} \subseteq \mathcal{C}$ and by the hypothesis of the lemma, it follows that

$$r \notin \bigcup_{C \in \mathcal{M}_{[r-\ell,r]}} \mathcal{I}(C).$$

Observe that

$$\bigcup_{C\in\mathcal{M}_{[r-\ell,r]}} \mathcal{I}(C) \subseteq [r-\ell, r).$$

Claim 26 states that the intervals of any two maximal clusters are disjoint, hence,

$$\sum_{C\in\mathcal{M}_{[r-\ell,r]}} |\mathcal{I}(C)| \leq \ell.$$

As $|\mathcal{I}(C)| = 2|C|/\delta_{\mathcal{TC}}$ for every $C \in \mathcal{C}$ and since $\mathcal{M}_{[r-\ell,r]}$ forms a partition of I, it follows that

$$\sum_{C\in\mathcal{M}_{[r-\ell,r]}} |\mathcal{I}(C)| = \frac{2k}{\delta_{\mathcal{TC}}}.$$

By the above two equations, we have that $\ell \geq 2k/\delta_{\mathcal{TC}}$. Substituting to Equation (8), we have that $\Delta((\mu_{r-\ell},\ldots,\mu_r),(\sigma_{r-\ell},\ldots,\sigma_r)) > 2k$. Since $\Delta((\tilde{\sigma}_{r-\ell},\ldots,\tilde{\sigma}_r),(\sigma_{r-\ell},\ldots,\sigma_r)) = k$, we have that $\Delta((\mu_{r-\ell},\ldots,\mu_r),(\tilde{\sigma}_{r-\ell},\ldots,\tilde{\sigma}_r)) > k$ in contradiction to Equation (7). ◀

Using the above, we obtain a better bound on the fraction of bad epochs compared to the bound $O(\varepsilon c/\delta_{\mathcal{TC}})$ established in Section 5.3.

▶ **Lemma 29.** *At most* $(4\varepsilon/\delta_{\mathcal{TC}} + \varepsilon(2c+2))$ *fraction of the epochs are bad.*

Proof. Observe that for every $C \in \mathcal{C}$ there exists a maximal cluster $M \in \mathcal{M}$ such that $C \subseteq M$. By Claim 24 it then follows that $\mathcal{I}(C) \subseteq \mathcal{I}(M)$, and so

$$\bigcup_{C\in\mathcal{C}} \mathcal{I}(C) = \bigcup_{M\in\mathcal{M}} \mathcal{I}(M).$$

Claim 27 implies that

$$\sum_{M\in\mathcal{M}} |\mathcal{I}(M)| \leq \frac{2\varepsilon R}{\delta_{\mathcal{TC}}}. \tag{9}$$

Notice that each cluster M intersect with at most $\lceil |\mathcal{I}(M)|/(c+1) \rceil$ bad epochs. By Claim 28, if $r \notin \mathcal{I}(M)$ for every $M \in \mathcal{M}$ then r is good. Hence there are at most

$$\sum_{M\in\mathcal{M}} \left\lceil \frac{|\mathcal{I}(M)|}{c+1} \right\rceil$$

bad epochs. Since the maximal clusters form a partition of $\mathcal{E}$, it follows that $|\mathcal{M}| \leq \varepsilon R$. This, together with Equation (9) yields

$$\begin{aligned}\sum_{M\in\mathcal{M}} \left\lceil \frac{|\mathcal{I}(M)|}{c+1} \right\rceil &\leq \varepsilon R + \sum_{M\in\mathcal{M}} \frac{|\mathcal{I}(M)|}{c+1} \\ &\leq \varepsilon R + \frac{2\varepsilon R}{\delta_{\mathcal{TC}}(c+1)} \\ &= \left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}} + \varepsilon(2c+2)\right) k.\end{aligned}$$

So, at most $(4\varepsilon/\delta_{\mathcal{TC}} + \varepsilon(2c+2))$ fraction of the epochs are bad as stated. ◀

By Claim 22 and Lemma 29 there are at least $(1-2(4\varepsilon/\delta_{\mathcal{TC}}+\varepsilon(2c+2)))\,k$ good epochs that are not short-split. By Claim 23, in each such epoch, Φ increases by at least $2c$. In the remaining epochs, Φ decreases by at most $6c$. Since $\Phi(-1)=0$ we have that

$$\begin{aligned}\Phi(R) &\geq \left(\left(1-2\left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}}+\varepsilon(2c+2)\right)\right)2c+2\left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}}+\varepsilon(2c+2)\right)\cdot(-6c)\right)k\\ &= \left(1-\frac{32\varepsilon}{\delta_{\mathcal{TC}}}-8\varepsilon(2c+2)\right)\cdot 2ck\\ &\geq \left(1-\frac{2}{c}-\frac{32\varepsilon}{\delta_{\mathcal{TC}}}-16c\varepsilon\right)R.\end{aligned}$$

By setting c to be an integer with $c=\Theta(\frac{1}{\sqrt{\varepsilon}})$ and $\delta_{\mathcal{TC}}=\sqrt{\varepsilon/\log(1/\varepsilon)}$, we get that

$$\Phi(R)\geq\left(1-\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})\right)R.$$

By setting $R=(1+\Theta(\sqrt{\varepsilon\log(1/\varepsilon)}))n$, and since $\mathcal{TC}$ has rate $1-\Theta(\delta_{\mathcal{TC}}\log(1/\delta_{\mathcal{TC}}))=1-\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})$, the first part of the scheme assures that $\mathsf{depth}(v(R))\geq n$. Similarly to the analysis of Part 2 from Section 5.3, by setting $\tau=\Theta(\mu)=\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})$, Theorem 6 follows.

References

1. S. Agrawal, R. Gelles, and A. Sahai. Adaptive protocols for interactive communication. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 595–599, 2016.
2. N. Alon, M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler. Reliable communication over highly connected noisy networks. In *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 165–173, 2016.
3. Z. Brakerski and Y. T. Kalai. Efficient interactive coding against adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–166, 2012.
4. Z. Brakerski, Y. T. Kalai, and M. Naor. Fast interactive coding against adversarial noise. *Journal of the ACM (JACM)*, 61(6):35:1–30, 2014.
5. Z. Brakerski and M. Naor. Fast algorithms for interactive coding. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 443–456, 2013.
6. M. Braverman. Towards deterministic tree code constructions. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 161–167, 2012.
7. M. Braverman and K. Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 236–245, 2014.
8. M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017.
9. M. Braverman and A. Rao. Towards coding for maximum errors in interactive communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 159–166, 2011.
10. M. Braverman and A. Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, 2014.
11. G. Cohen, B. Haeupler, and L. J. Schulman. Explicit binary tree codes with polylogarithmic size alphabet. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 535–544. ACM, 2018.
12. R. Gelles. Coding for interactive communication: A survey. *Foundations and Trends in Theoretical Computer Science*, 13(1-2):1–157, 2017. doi:10.1561/0400000079.
13. R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson. Towards optimal deterministic coding for interactive communication, 2016.

14 R. Gelles, A. Moitra, and A. Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.

15 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 768–777. IEEE, 2011.

16 M. Ghaffari and B. Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, 2014.

17 M. Ghaffari, B. Haeupler, and M. Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 794–803, 2014.

18 B. Haeupler. Interactive Channel Capacity Revisited. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 226–235, 2014.

19 A. Jain, Y. T. Kalai, and A. B. Lewko. Interactive coding for multiparty protocols. In *Proc. ACM Innovations in Theoretical Computer Science Conference (ITCS)*, pages 1–10, 2015.

20 G. Kol and R. Raz. Interactive channel capacity. In *STOC*, volume 13, pages 715–724, 2013.

21 E. Kushilevitz and N. Nisan. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

22 A. Lewko and E. Vitercik. Balancing communication for multi-party interactive coding. *CoRR*, abs/1503.06381, 2015. `arXiv:1503.06381`.

23 C. Moore and L. J. Schulman. Tree codes and a conjecture on exponential sums. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 145–154, 2014.

24 Anand Kumar Narayanan and Matthew Weidner. On decoding Cohen-Haeupler-Schulman tree codes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1337–1356. SIAM, 2020.

25 P. Pudlák. Linear tree codes and the problem of explicit constructions. *Linear Algebra and its Applications*, 490:124–144, 2016.

26 S. Rajagopalan and L. J. Schulman. A coding theorem for distributed computation. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 790–799, 1994.

27 A. Rao and A. Yehudayoff. Communication complexity (early draft), 2018.

28 L. J. Schulman. Communication on noisy channels: a coding theorem for computation. *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1992.

29 L. J. Schulman. Deterministic coding for interactive communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 747–756, 1993.

30 L. J. Schulman. Postscript of 21 September 2003 to Coding for Interactive Communication. http://users.cms.caltech.edu/∼schulman/Papers/intercodingpostscript.txt, 1994.

31 L. J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

32 A. A. Sherstov and P. Wu. Optimal interactive coding for insertions, deletions, and substitutions. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017.