Locally Testable Tree Codes

Tamer Mour * Alon Rosen† Ron D. Rothblum‡

April 29, 2024

Abstract

Tree codes, introduced in the seminal works of Schulman (STOC 93', IEEE Transactions on Information Theory 96') are codes designed for interactive communication. Encoding in a tree code is done in an online manner: the i-th codeword symbol depends only on the first i message symbols. Codewords should have good *tree distance* meaning that for any two codewords, starting at the first point of divergence, they should have large Hamming distance.

We investigate whether tree codes can be made to be *locally testable*. That is, can a tester, which is given oracle access to an alleged codeword w of the tree code, decide whether w is indeed a codeword or far from such, while only reading a sub-linear number of symbols from w.

As the main result of this work, we construct, for any $r \geq 3$, a *probabilistic* tree code that is locally testable using $\tilde{O}(n^{2/r})$ queries. The tester accepts any codeword with probability 1 and rejects strings that are δ_r-far from the code with high probability, where $\delta_r < 1$ degrades with r. Our probabilistic notion of a tree code is a relaxation of the standard notion and allows the encoder to toss random coins. We require that encoded messages are far (in tree distance) from any possible encoding of any other message.

*Bocconi University, BIDSA. E-mail: tamer.mour@unibocconi.it. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547).

†Bocconi University, BIDSA, and Reichman University. E-mail: alon.rosen@unibocconi.it. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

‡Technion. E-mail: rothblum@cs.technion.ac.il. Supported by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction

Suppose we wish to encode a huge amount of data that is created across multiple generations. Each generation creates new information and we would like to encode it in an online manner into a repository. Naturally, we want the encoding to be such that in case a few errors happen we can still decode and retrieve the information. A solution to this problem is offered by the beautiful notion of a *tree code*, proposed by Schulman [Sch93, Sch96].

Tree codes, originally designed for encoding interactive communication, have two distinctive features:

1. The encoding is done in an online manner. This means that the i-th symbol c_i of the codeword, can depend only on the first i symbols of the message (and not on future message symbols, that in our context will be created by a future generation).

2. Given that codeword symbols cannot depend on future message symbols, one cannot hope to achieve the standard notion of large Hamming distance between distinct codewords. Rather, in tree codes we aim to have the property that once two codewords diverge, they should be distinct on many coordinates. This guarantees that once an error happens, hereon, unless a very large number of errors should happen, we can still recover the data.

In his seminal work, Schulman [Sch93, Sch96] showed that tree codes exist, albeit non-explicitly. Specifically, based on a probabilistic argument, he proved that there exist tree codes with constant rate and constant distance. In a following postscript [Sch94], he additionally describes an explicit construction of a tree code that has inverse logarithmic rate. A recent exciting work by Cohen, Haupler and Schulman [CHS18] gives an explicit construction with an exponential improvement in the rate, i.e. with rate $\Theta(1/\log\log(n))$.

Returning to our cross generational encoding scheme, given the huge size of the database, we want to enable testing that the stored data is still properly encoded. That is, to very efficiently test that not too many errors occurred. In the standard coding setting, this can be achieved via *locally testable codes* (LTCs). An LTC is a standard error-correcting code (i.e., distance is measured according to standard Hamming distance), which additionally offers a method to efficiently detect whether a given string is *far* from the code, while reading only a few of the bits. To the best of our knowledge, no tree code in the literature is known to be locally testable.

LTCs grew out of the PCP literature, and indeed form the heart of many PCP constructions. The notion originates in the works of Blum, Luby and Rubinfeld [BLR93] on linearity testing and Rubinfeld and Sudan [RS96] on low degree testing. LTCs were fully formulated by Goldreich and Sudan [GS06]. Following a long body of work [GS06, BSVW03, BGH+06, BS08, Din07, KMRS17], in a recent exciting breakthrough, Dinur *et al.* [DEL+22] constructed an LTC with constant rate, constant relative distance and constant query complexity. Still, all existing constructions of LTCs are with respect to the Hamming distance metric, and in particular do not allow for an online encoding as in tree codes. As such, they are unsuitable for our application.

1.1 Locally Testable Tree Codes

Our main result is a construction of a variant of tree codes, which is locally testable. In our variant, and in contrast to the usual setting in tree codes, the encoding function can be *randomized*. Before defining the precise model, we need to introduce the notion of *tree distance* and the corresponding notion of a (standard) *tree code*.

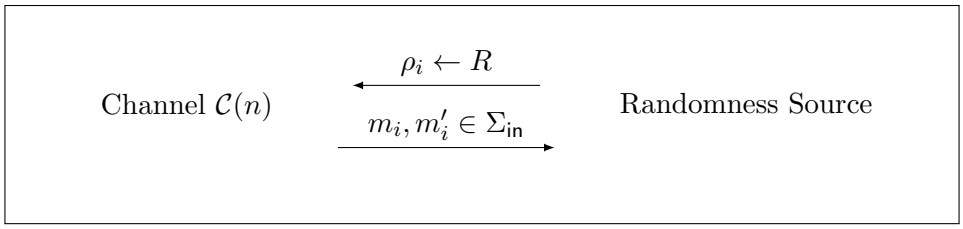

Figure 1: The channel model for Definition 1.3.

Definition 1.1 (Tree Distance)**.** *Let Σ be an alphabet and $n \in \mathbb{N}$. Let $w, w' \in \Sigma^n$ and let $i^* = \min\{i : w_i \neq w'_i\}$ (and $i^* = 0$ when $w = w'$). We define the* tree distance *between w and w' as*

$$\Delta_{\mathsf{T}}(w, w') = \Delta_{\mathsf{H}}(w_{\geq i^*}, w'_{\geq i^*}),$$

where Δ_{H} denotes the relative Hamming distance and $w_{\geq i^*}$ the suffix of w starting at position i^*.

Definition 1.2 (Tree Code)**.** *A* tree code *is an injective function $C = \left\{C_n : (\Sigma_{\mathsf{in}})^n \to \Sigma_{\mathsf{out}}\right\}_{n\in\mathbb{N}}$ which, on input $m = (m_1, \ldots, m_n) \in (\Sigma_{\mathsf{in}})^n$, outputs*

$$C(m) = \big(C_1(m_1), C_2(m_1, m_2), \ldots, C_n(m)\big) \in (\Sigma_{\mathsf{out}})^n.$$

We say that C has distance δ *if for every distinct $c, c' \in C$ of the same length, it holds that $\Delta_{\mathsf{T}}(c, c') \geq \delta$.*

Here and throughout, we allow the output alphabet to grow as a function of n. In such a case, Σ_{out} may be viewed as an ensemble $\Sigma_{\mathsf{out}} = \left\{\Sigma^{(n)}_{\mathsf{out}}\right\}_{n\in\mathbb{N}}$, where $\Sigma^{(n)}_{\mathsf{out}}$ is the alphabet for the n^{th} codeword symbol, output by C_n. The *rate* of the code is then defined as $\log(|\Sigma_{\mathsf{in}}|)/\log(|\Sigma^{(n)}_{\mathsf{out}}|)$.

As mentioned above, we allow the encoding function to be probabilistic. Similarly to a tree code, our code works in an online manner – in each round we are given a new message symbol and in addition some new randomness, and we output a single codeword symbol (that may additionally depend on the prior message and randomness symbols but not on the future ones).

We require encodings of distinct messages to be far from one another, under the tree distance, even under a very strong adversarial model. The adversary, which we think of as a "channel", can select the message bits as a function of all previously generated random bits, including ones generated for the current symbol.

Thus, we consider a channel $\mathcal{C}$ that operates as follows. In round i, a random symbol ρ_i is selected (from some randomness space R) and seen by the channel. Then, the channel selects two message symbols m_i, m'_i (which are not necessarily distinct). The channel terminates after n rounds and so, overall, we obtain (ρ, m, m'), where $\rho = (\rho_1, \ldots, \rho_n)$, $m = (m_1, \ldots, m_n)$ and $m' = (m'_1, \ldots, m'_n)$. We further require the channel to be such that the two generated messages m and m' are distinct. A depiction of the channel interaction is provided in Fig. 1.

Definition 1.3 (Probabilistic Tree Code)**.** *A* probabilistic tree code *$C = \{C_n : \Sigma^n_{\mathsf{in}} \times R^n \to \Sigma_{\mathsf{out}}\}$ is a tree code where the encoding function takes in randomness. Specifically, the encoding of the message $m = (m_1, \ldots, m_n) \in \Sigma^n_{\mathsf{in}}$ under randomness $\rho = (\rho_1, \ldots, \rho_n) \in R^n$, is defined as*

$$C(m; \rho) = \big(C_1(m_1; \rho_1), C_2(m_1, m_1; \rho_1, \rho_2), \ldots, C_n(m; \rho)\big).$$

We say that C has tree distance $\delta : \mathbb{N} \to [0,1]$*, denoted* $\Delta_{\mathsf{T}}(C)$*, if for any channel* $\mathcal{C}$ *that always outputs distinct messages* $m \neq m'$*, it hold that*

$$\Pr_{(m,m',\rho)\leftarrow\mathcal{C}(n)}\Big[\exists \rho' \in R^n \text{ s.t. } \Delta_{\mathsf{T}}\big(C(m;\rho), C(m';\rho')\big) < \delta(n)\Big]$$

is a negligible function in n*.*

In our constructions we have that $\Sigma_{\mathsf{out}} = (\Sigma_{\mathsf{in}})^{\ell}$ for some parameter $\ell = \ell(n)$. In such a case notice that the rate is $1/\ell$.

Note that our notion of distance for a probabilistic tree code is that the encoding, under randomness ρ of the message m, should be far (in tree distance) from the encoding of m' *under any possible randomness* ρ'. This means that once a message is encoded, as long as few errors occur, it is close in tree distance to a *unique* codeword *defined under any randomness.*

We say that a (possibly, probabilistic) tree code C is locally testable with q queries for distance δ, if there exists a randomized algorithm that makes at most q queries, accepts any string $c \in C$ with probability 1, and rejects any string that is δ-far from C, in tree distance, with probability at least $1/2$.

1.2 Main Result

Our main result, stated in the theorem below, is the construction of a locally testable probabilistic tree code. We refer the reader to Corollary 7.5 for a more detailed statement.

Theorem 1.4 (Main Result)**.** *For any* $r \geq 3$*, there exists a locally testable probabilistic tree code* C *with rate* $1/\mathrm{polylog}(n)$ *and distance* $1 - o(1)$*. Specifically, there exists a tester for* C *that makes* $\tilde{O}(n^{2/r})$ *queries and rejects any word that is* $(1 - H_r/r + o(1))$*-far from the code.*

Here, $H_r = \sum_{i=1}^{r} 1/i$ is the r^{th} harmonic number, which satisfies that $H_r \to \ln(r) + \gamma$, where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. In particular, an instantiation of the theorem with $r = 3$ gives a local test that rejects words that are far from the code yet are close enough to be in the unique-decoding radius of some codeword (this is because $1 - H_3/3 = 14/36 < 1/2 - o(1)$). As r increases, the query complexity of the tester improves, but the "testing radius" (i.e., the distance in which the tester is guaranteed to reject) degrades and, in fact, approaches 1. An intriguing question that we leave open is whether we can build a tree code that is a *strong* LTC; namely, where the test rejects *any* δ-far word with probability $\Omega(\delta)$.

Theorem 1.4 is proved by developing a generic transformation that takes any tree code, satisfying a certain linear structure, and transforms it into a (probabilistic) locally testable tree code. We instantiate our transformation with the tree code constructed by Evans, Klugerman and Schulman [Sch94], specifically, a variant thereof that is presented in [Gel17, Section 3.1.1]. Their construction is explicit and yields a tree code with constant distance and polylogarithmic rate, using an error-correcting block code[1] with constant distance and constant (or even polylogarithmic) rate, e.g., Reed-Solomon. In addition, and crucially to us, their code satisfies our special notion of linearity whenever the underlying block code is linear.

Since the introduction of tree codes and their first constructions [Sch93, Sch94, Sch96], the tree code literature is mainly interested in minimizing the rate of tree codes, namely the size

[1] Recall a block code is the "standard" notion of an error correcting code, in particular where the length of codewords is apriori fixed.

of their output alphabets. The "holy grail" is building tree codes with constant distance over constant-size alphabets, with the state-of-the-art consisting of *non-explicit* tree codes that achieve constant rate [Sch96], or *explicit* tree codes that have rate $\Theta(1/\log\log(n))$ (with n being the length of the encoded message thusfar) [CHS18]. Notably, the latter construction builds on a "rate-1/2" construction over the integers. However, constructions with infinite alphabets are practically incomparable to the setting where the alphabets are limited to be finite.

In contrast, our focus is to achieve tree codes that are locally testable. When local testability is not required, there is a simple and generic construction of tree code from good error-correcting codes that has rate $1/\log(n)$ (i.e. polynomial-size alphabet) [Sch94]. As far as we know, however, this construction does not preserve local testability. In particular, we are not aware of any direct or indirect construction of a locally testable tree code with any non-trivial rate.

Remark 1.5. *One can instantiate our transformation using the non-explicit constant-rate tree code from [Sch96]. However, the resulting code will still have rate* $1/\mathrm{polylog}(n)$ *due to an overhead incurred by our transformation. We also remark that the code from [CHS18] with rate* $\Theta(1/\log\log(n))$ *is not linear and is therefore inapplicable to our purpose.*

1.3 Towards Tree PCPs?

Our question of encoding information across generations stems from a more complex question of similarly encoding *computations*. We think of an enormous computational task carried over centuries by different generations. We would like to be able to encode the computation that was performed so far in such a way that it is always possible to quickly check that it was performed correctly.

We envision a construction of a *tree PCP*: a proof of correctness that can be generated in an online manner and verified using a few queries. Given that LTCs form the heart of many PCP constructions, we view our construction of a (probabilistic) locally testable tree code, as the first step in such a construction.

We remark that this notion of a tree PCP, is related to, but distinct from, *incremental PCPs* proposed by Naor, Paneth and Rothblum [NPR19]. In an incremental PCP, the PCPs symbols all change as the computation evolves but they do so separately. That is, each symbol "updates itself" without looking at any of the other symbols. In contrast, in a tree PCP whatever part of the proof has already been generated becomes canon and any newly generated proof is appended to the existing proof (and is therefore amenable to being constructed on, say, a blockchain).

A different approach for verifying an extremely long computation is via *incrementally verifiable computations* (IVC) proposed by Valiant [Val08] (see also [BCCT13, PP22]). An IVC is a cryptographic proof-system in which one can prove, via a succinct proof, that a long computation was done correctly. Moreover, IVCs require that the proof can be very efficiently updated as the computation proceeds (in time independent in the length of the computation thusfar). In contrast, we seek an information theoretic analog of tree codes for PCPs, where soundness is statistical but we allow the update time to grow with the length of computation.

1.4 Techniques and Ideas

Here, we give a bird's eye view of the technical contributions of this work. A more detailed overview is provided in Section 2.

The starting point for our construction is *code tensoring*, which has proven to be an extremely effective tool to obtain local testability in the standard setting [BS04, Mei09, Vid15, KMRS17]. Most importantly for our purpose, Viderman [Vid15] shows that, for $r \geq 3$, the r-fold tensor of *any* code with good distance is locally testable.

Similarly to the standard notion (i.e. tensor of a block code), we define the r-fold tensor of a tree code C, which we denote by C^r, to be the code that consists of all r-dimensional tensors where every (axis-parallel) line is a codeword. While tensor tree codes do not immediately give locally testable tree codes (as we discuss later on), they already give us meaningful properties. (i) First, just by definition, they guarantee tree distance at any restriction to a line. (ii) Second, it turns out that when the base code C is a special case of a linear tree code, namely a *linear vector tree code*, then there exists an encoding function for C^r that is "online in r dimensions".[2] Specifically, the codeword symbol at location $\mathbf{i} = (i_1, \ldots, i_r)$ can be computed given all message symbols at locations $\mathbf{i}' = (i'_1, \ldots, i'_r)$ where $i'_j \leq i_j$ for all j (in such a case we say that $\mathbf{i}' \leq \mathbf{i}$). (iii) Lastly, by adapting the proof from [Vid15] to our setting, we show that tensor tree codes are locally testable for tree distance at all lines. We elaborate on this below.

Roughly speaking, the test from [Vid15], which originates in [BS04], tests tensor codes for Hamming distance by checking the codeword at a uniformly random $(r-1)$-dimensional hyperplane. In the context of tree distance, the uniform distribution is unsuitable (indeed, it will not place enough weight on the suffix) and so we modify the distribution by which the test samples the hyperplane to give more weight to hyperplanes closer to the end. Intuitively speaking, since tree distance counts the fraction of disagreements only in the suffix starting at the divergence, hyperplanes that are closer to the end have more impact on tree distance simply because they appear in more suffixes (in contrast to Hamming distance, where all hyperplanes have equal impact).

We show that such a test gives local testability even for a weaker notion of distance, which we call *suffix distance*. At a high level, suffix distance between two words is defined as the probability of disagreement when sampling a random point according to a distribution that is skewed towards the end, similarly to what is described above.

So far we have described how to build tensor tree codes and identified their potentially useful features. Unlike the standard setting, however, more work is needed to obtain tree LTCs. Recall that tensors of block codes are useful since, in particular, they immediately convert into block codes; Any embedding of the high-dimensional codewords into one dimension (which we hereby refer to as *flattening*) preserves the Hamming distance. Further, the encoding function (when the base code is linear) translates into an encoding function for the flattened code, and so does the local test. In contrast, it is not clear whether a tensor tree code can be flattened to build a tree code.

To begin with, in order to get the online encoding aspect in our tree code, the embedding of the r-dimensional coordinate space into one dimension must be *monotone*. That is, a coordinate $\mathbf{i}$ must be embedded *after* all coordinates $\mathbf{i}' \leq \mathbf{i}$ to allow for online encoding. While there are plenty of ways to define such a monotone embedding, we must be careful that our embedding preserves the distance and local testability of the tensor tree code.

To that end, we devise an embedding that follows a traversal of the r-dimensional space, such that the n^{th} coordinate that is reached by the traversal is embedded at location t on the one-dimensional line. At a high-level, our traversal proceeds in layers where, at the t^{th} layer, it covers

[2] Recall that, even in the case of block codes, it is unclear how to encode a message under a tensor code unless the base code is linear. In fact, the tensor of non-linear block codes might be empty.

all coordinates $\mathbf{i}$ such that $L_\infty(\mathbf{i}) = t$. That is, by the end of the t^{th} layer, the traversal covers all coordinates in the sub-cube ending at coordinate $\mathbf{t} = (t, \ldots, t)$. Inside each of these layers, which can be decomposed into segments of lower dimension, the traversal behaves recursively.

The key feature of our traversal is the following: The set of coordinates that the traversal reaches up to any point can be covered by the union of a constant number of rectangular tensors. In other words, the embedding of any (one-dimensional) codeword back to the r-dimensional space produces a partial tensor that has a constant number of *endpoints*, i.e. maximal coordinates.[3] Roughly speaking, such a bound is crucial to us since these endpoints essentially correspond to parts of our flattened codeword (or, the partial tensor it embeds to) where we expect to have codewords of the tensor tree code (recall codewords in the tensor tree code are, by definition, rectangular). We elaborate below.

First, we show that the flattened code can be locally tested by applying the local test for the tensor tree code at a randomly selected endpoint; By thorough analysis, we show that if the partial tensor is far from the tensor tree code in tree distance, then there must exist an endpoint where the corresponding rectangle is far from the code in suffix distance. This implies local testability due to the local testability of the tensor tree code to suffix distance.

Second, we show that tree distance at all lines, guaranteed by the tensor code, can be bootstrapped to tree distance of the flattened code using randomness. Specifically, we build a probabilistic tree code using the tensor tree code that, when encoding the n^{th} codeword symbol, additionally samples a bunch of prior symbols and duplicates them, with the goal of "dragging past errors". The symbols that are sampled at time n concentrate around the endpoints of the partial tensor of size n; We show that, although tree distance at all lines in the tensor code does not necessarily imply tree distance in its flattening, it does imply suffix distance at one of the endpoints. Hence, when properly sampling random symbols around the endpoint, we expect to amplify the distance and produce many disagreements after the first divergence (with overwhelming probability).

1.5 On the Probabilistic Model

As noted above, our construction achieves a probabilistic notion of tree distance. Recall that a probabilistic tree code is a tree code that guarantees distance in an adversarial noisy channel. Our probabilistic model differs from the standard model of tree codes in the following two aspects. First, we allow the online encoding algorithm to use randomness. Notice this randomness does not play any part in the decoding and, therefore, should be thought of as local randomness and need not be communicated. Second, we limit the adversarial channel to choose its corruptions in an online manner; That is, the noise at any point in time cannot depend on the randomness sampled for encoding symbols in the future. In an intuitive sense, we impose a "fair game" between the communicating parties (the encoder and decoder) and the adversarial channel: On the one hand, similarly to the standard notion, the encoder must encode any symbol independently of future symbols and future noise. On the other hand, the channel must choose its corruption of any symbol independently of future encoding randomness.

Consequently, we believe that probabilistic tree codes are a natural relaxation of tree codes, that can be used to replace the standard notion in any natural application where local randomness is allowed. In particular, similarly to the standard notion [Sch93, BR14], probabilistic tree codes can

[3] More formally, we say that a coordinate $\mathbf{e}$ is an endpoint of a partial tensor if, for any coordinate $\mathbf{i}$ populated by the partial tensor, it holds $\mathbf{i} \not\succeq \mathbf{e}$.

be also used to construct interactive encoding schemes whenever the adversarial channel is assumed to act in the above online manner. For instance, this aligns with the natural real-life scenario where the channel may be arbitrarily noisy, yet once a message that is sent by Alice reaches Bob, it is recorded and is no longer corruptible by the channel. Further, only after recording Alice's message, Bob sends the next message in the protocol (and encodes it using fresh randomness).

2 Technical Overview

Our goal is to construct locally testable probabilistic tree codes. These are codes that (i) can be encoded in an online fashion, (ii) exhibit good tree distance, and (iii) are accompanied with a tester that, by looking at few locations in a word, is able to test its proximity to the code.

2.1 Local Testability via Tensoring

An important tool for generically constructing locally testable codes is the *tensoring* operation (see, e.g., [BS04, Mei09, Vid15, KMRS17]). Let $\mathbb{F}$ be a field. The tensor product of two codes $C_1 \subseteq \mathbb{F}^{n_1}$ and $C_2 \subseteq \mathbb{F}^{n_2}$, denoted $C_1 \otimes C_2 \subseteq \mathbb{F}^{n_1 \times n_2}$, consists of all $n_1 \times n_2$ dimensional matrices where every column is a codeword of C_1 and every row is a codeword of C_2. In this work, we will always tensor a code C with itself, possibly $r > 2$ times, which results in the r-fold tensor $C^r = C \otimes \cdots \otimes C$.

While tensoring incurs a tolerable loss in distance (when C has relative distance δ, C^r guarantees distance δ^r), it grants us the local testability we seek. Due to the strong structural properties of tensor codes, it turns out that one can look at very few locations in a tensor to determine, with good probability, whether it is in the code or whether it is far from it. A typical tester for a tensor code is in fact very simple. In particular, a natural tester for the 2-fold tensor product C^2 simply consists of sampling a line over the coordinates of the input matrix and checking whether the restriction of the input codeword to the line belongs to the base code C.

Somewhat surprisingly, Valiant [Val05] (see also [GM12]) showed the above test does not always work but an analogous test for dimension $r \geq 3$ does work. Indeed, Viderman [Vid15] gave a general local testability theorem for tensor codes. He showed that, for $r \geq 3$, the r-fold tensor of *any* code C (with good distance) is locally testable. For the sake of this outline, let us therefore fix $r = 3$. The test is a generalization of the test sketched above to 3 dimensions: To test a three-dimensional tensor, sample a random two-dimensional plane[4] and test its membership in C^2. The power of Viderman's result stems from its generality: his analysis works for *any* tensor code. Thus, given an arbitrary code, using Viderman's result, we can compose it with itself and derive an LTC.

2.2 Tensoring Tree Codes

A natural approach would be to try and achieve local testability for tree codes via tensoring. Towards this end, we think of a tree code C as an *infinite collection of block codes* $\{C_n\}_{n\in\mathbb{N}}$, where, for any $n \in \mathbb{N}$, C_n is a block code with block-length n, in which any codeword has an $(n-1)$-prefix from C_{n-1}. Given a tree code C, with its corresponding collection of block codes $\{C_n\}_{n\in\mathbb{N}}$, we may define the tensor code $C^2 = C \otimes C$ to consist of all codewords that are in $C_{n_1} \otimes C_{n_2}$ for some $n_1, n_2 \in \mathbb{N}$. In other words, C^2 is the infinite collection of tensor block codes $\{C_{n_1} \otimes C_{n_2}\}_{n_1,n_2\in\mathbb{N}}$.

[4] While such a tester has query complexity $n^{2/3}$, in the general case where $r > 3$, one can generically reduce the complexity of the test to $n^{2/r}$ via recursion (see Proposition 3.11).

This, in turn, can be generalized to any order $r > 2$ to capture the r-fold tensor of of a tree code C, which we denote by C^r.

It is not at all clear whether tensor tree codes give us anything meaningful, let alone whether they correspond to the standard notion of a tree code. To gain better intuition, let us take a closer look on the properties of such a tensor code C^r for $r = 2$, ignoring for the moment the aforementioned limitations on this case in the standard setting [Val05, GM12].

Online Encoding in Two-dimensions. First, observe that by our definition, any codeword $\mathbf{c}$ of length $n_1 \times n_2$ in the tensor code C^r consists of a shorter codeword of length $n_1 \times (n_2 - 1)$ and, simultaneously, a shorter codeword of length $(n_1 - 2) \times n_2$ (obtained by removing the last column or, resp., last row). In fact, the restriction of $\mathbf{c}$ to all coordinates except for the furthest corner, namely $([n_1] \times [n_2]) \setminus \{(n_1, n_2)\}$, is a codeword in the corresponding restriction of the code. This gives hope for some notion of online encoding since a similar structure, though over the one-dimensional line, is essentially what allows for a tree code to be encoded in an online manner.

Even in the standard setting, it is generally not clear how to build an encoding function for a tensor C^2 given an encoding function of C. Fortunately, however, when we assume that the base code C is *linear*, then this task is much simpler thanks to the algebraic structure of linear codes and their tensor products. More specifically, letting $C : \mathbb{F}^k \to \mathbb{F}^n$ be a linear code,[5] a possible encoding function of the tensor C^2 takes as input a matrix $\mathbf{m} \in \mathbb{F}^{k \times k}$, applies C over its rows to obtain an $k \times n$ matrix $\mathbf{c}'$ over $\mathbb{F}$, then applies C over the columns of $\mathbf{c}'$ to obtain the final codeword $\mathbf{c} = C^2(m) \in \mathbb{F}^{n \times n}$. In particular, it holds that such an encoding function corresponds to the linear map $C \otimes C$ (here, $\otimes$ is the tensor product operation over linear maps).

We propose a notion of *linear vector tree codes* that enables us to follow an outline similar to the above. Specifically, we require that the linear tree code takes as input message symbols from a field $\mathbb{F}$ and that any codeword symbol is a vector over $\mathbb{F}$ obtained as a linear transformation on the message thusfar. Thus, the output alphabet is a vector space over the input alphabet, which is itself a field. While this is often regarded as the standard notion of linear block codes, things are different in the tree code literature (e.g. [CHS18]). Specifically, since, by syntax, a tree code inherently maps n input symbols to n output symbols, the output alphabet must be larger than the input alphabet to obtain redundancy. Therefore, linear tree codes are generally defined as mapping one ring to another. Our definition of linear vector tree codes further imposes the above vector-space structure, which does not necessarily hold in general linear tree codes. A similar definition to ours is made in [Pud13], though it considers the less general case of finite tree codes.

The linear vector notion essentially means that the message space and codeword space, for any fixed length $n \in \mathbb{N}$, are both vector spaces over the same field. Consequently, any linear vector tree code C can be viewed as a collection of linear block codes $\{\mathbf{C}_n : \mathbb{F}^n \to \mathbb{F}^{\ell \cdot n}\}$ for some $\ell \in \mathbb{N}$, which possibly grows with n.

An example of such a linear vector tree code is in one of Schulman's [Sch96] original (non-explicit) tree code construction (as presented in [Gel17, Section 2.1]). That construction encodes binary messages while utilizing a random polynomial, which can be fixed non-uniformly (and is the reason the construction is non-explicit). If that polynomial is over an extension field of the binary field (i.e., it has characteristic 2) then the resulting construction is a linear vector tree code. Similarly, one of the main constructions in [CHS18] (namely, the one directly based on Newton

[5] Note we overload notation and use C to denote both the code and the encoding function.

polynomials) has similar structure, albeit over the integers. Thus, in the sequel, we will assume that our base tree code is indeed a linear vector tree code.

Interpreting any linear vector tree code C as a collection of (standard) linear codes that have an online encoding function, we are able to identify an encoding function for the tensor C^2 (and, more generally, for C^r for any $r > 1$). The encoding function takes as input a matrix $\mathbf{m} \in \mathbb{F}^{n_1 \times n_2}$ and outputs a codeword $\mathbf{c} \in \mathbb{F}^{\ell n_1 \times \ell n_2}$. The "online-ness" of the encoding of C translates to the following notion of online encoding for C^2: To encode the symbol at location $\mathbf{i} = (i_1, i_2)$ in $\mathbf{c}$, it is sufficient to look only at the input symbols at coordinates $\mathbf{i}' = (i'_1, i'_2)$ such that $i'_1 \leq i$ and $i'_2 \leq i_2$. Thus, one can think of this encoding function as an online function with respect to a "two-dimensional time", where only a partial order between coordinates exist. More specifically, letting $\mathbf{i}' \leq \mathbf{i}$ denote the case where $i'_1 \leq i_1$ and $i'_2 \leq i_2$, then our encoding function outputs the "next symbol" at time $\mathbf{i} \in \mathbb{N}^2$ (i.e. the symbol at location $\mathbf{i}$ in the codeword) given all message symbols from time $\mathbf{i}' \leq \mathbf{i}$ – intuitively speaking, all symbols from the "two-dimensional past".

We refer to a code that satisfies the above online encoding notion as a *two-dimensional tree code*. In the general case, we use the term *high-dimensional tree code*.

Keep in mind that, while we have identified an encoding function for tensor tree codes that satisfies a meaningful "online-ness" notion, it is not clear how to convert such an encoding function to a properly-online encoding function as required by a tree code; Say we want to build a tree code given a high-dimensional tree code. It is not even clear what constitutes the encoding of the i^{th} codeword symbol given i input symbols, under such a transformation.[6] Let us put this issue aside for now and continue our investigation in the other desired properties in tensor tree codes, namely distance and local testability.

Distance at all Lines. When it comes to distance, it is immediate by definition that any restriction of C^2 to a line (either a row or a column) yields a tree code (which is in fact C). Hence, for any two codewords $\mathbf{c}, \mathbf{c}' \in C^2$ of length $n_1 \times n_2$, and any line ℓ (of length n_1 or n_2), it holds that, if $\mathbf{c}$ and $\mathbf{c}'$ disagree at some point on ℓ, then their respective restrictions to ℓ must be far in tree distance. A high-dimensional code that satisfies this property and, in particular, any C^r for any $r > 1$, is said to have *tree distance at all lines*. Following up on the intuition from above, one can think of this distance notion as the analog of tree distance in the imagined high-dimensional time, where tree distance holds across any one-dimensional timeline.

2.3 Testing Tensored Tree Codes

Recall that Viderman [Vid15] showed that any tensor code C^r is an LTC given C has good Hamming distance and $r \geq 3$. It would be useful if we could prove an analogous theorem for tensor tree codes. Specifically, local testability with respect to the distance notion they guarantee, namely tree distance at all lines.

The tester from [Vid15] for, say, the 3-dimensional tensor C^3, samples a uniformly random two-dimensional (axis-orthogonal) plane and checks whether it is in C^2. Such a test cannot possibly

[6] Note that the notion of a block code generalizes naturally to include codes over high-dimensional spaces; An r-dimensional block code C is a subset of codewords over some Σ of length $N = n_1 \times \cdots \times n_r$, i.e. $C \subseteq \Sigma^N$. This generalization is immaterial to the standard distance metric in code theory, namely the Hamming distance, as the latter is oblivious to the number of dimensions in a code's domain; the codewords of any high-dimensional code can be "flattened" to one dimension preserving the Hamming distance. As demonstrated, however, there is a more substantial gap between the notion of a tree code and its high-dimensional variant.

be useful to detect high tree distance at all lines; For simplicity, assume we have a code with tree distance $1/2$ at all lines and that we want to test for distance $1/3$. Consider a word $\mathbf{w} \in \Sigma^{n\times n\times n}$ where the only non-zero value is at coordinate $(n-2, n-2, n-2)$. Such a word cannot possibly be a codeword since it is $1/3$-close to the all zero string (recall the code is linear and so the all zero string is always a codeword). However, the test will sample a non-zero plane only with probability only $1/n^2$. Hence, roughly put, the uniform distribution over planes is simply not suitable for detecting tree distance.

The above observation leads us to identify a distribution over the planes that suits our purpose. Along the way and, in particular, to facilitate our analysis, we define a new notion of distance called *suffix distance*.

The Suffix Distribution. Let us look again at the toy example from above. For simplicity, let us consider a simpler variant, where the goal is to "catch" a non-zero in a one-dimensional codeword w with tree weight δ (i.e. tree distance δ from the zeros string). Intuitively, sampling a uniformly random point is natural for testing Hamming distance since, if w had *Hamming* weight δ, then a uniformly random point is a non-zero with probability δ. Let us try, then, the following naive solution for detecting tree weight: First, "guess" a location j at which we suspect the first disagreement occurs. Second, sample a uniformly random point in the suffix starting at j. Since tree distance is essentially the Hamming weight after the first disagreement, this test will catch a non-zero with probability at least δ/n. Unfortunately, this is also (almost) the best we can do using such a test, with the worst case being when the first disagreement happens almost at the very end (in which case the gap between tree weight and Hamming weight is the largest).

A better idea that avoids the linear loss is to try to guess where the first non-zero occurs up to a factor 2 multiplicative approximation. That is, sample $j \in [\log(n)]$ uniformly at random, and observe that, with probability at least $1/\log(n)$, the first non-zero falls in the interval $[n - 2^j, n - 2^{j-1} + 1]$. Notice that if we guessed correctly, then, due to the tree distance, the interval $[n - 2^j, n]$ must contain at least a $\delta/2$ fraction of non-zeros. Thus, by sampling $i \in [n - 2^j, n]$ uniformly at random, with probability at least $\delta/2$ we will find a non-zero. Overall, we catch a non-zero with probability at least $\frac{\delta}{2\log n}$.

Notice that the distribution described above for sampling an index i is heavily skewed towards the end of the string. Indeed, a simple calculation shows that a particular index i is sampled with probability roughly

$$\frac{1}{\log n} \cdot \sum_{\ell=\log(n-i)}^{\log(n)} \frac{1}{2^\ell} \approx \frac{1}{\log(n)} \cdot \frac{2}{n-i}.$$

The above gives us an inspiration to define a "nicer" distribution, that can be thought of as a "smoothening" of the test above. We define the *suffix distribution* over a range $[n]$, to sample an integer $i \in [n]$ with probability

$$\sigma_n(i) = \frac{1}{H_n} \cdot \frac{1}{n-i}, \tag{1}$$

where H_n is the n^{th} harmonic number[7] and its inverse is used for normalization (i.e., so that σ_n indeed specifies a distribution over $[n]$).

[7] $H_n = \sum_{i=1}^{n} 1/i$. It holds that $H_n = \ln n - O(1)$

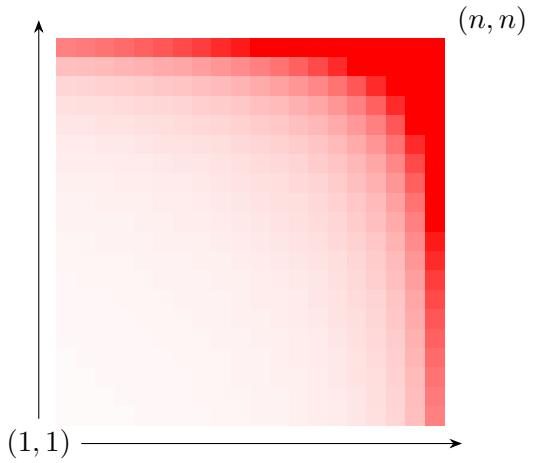

Figure 2: The suffix distribution over two dimensions.

The Test. It turns out that the suffix distribution is a key to converting the usefulness of tensors in standard LTCs to the tree code settings. We propose the following test for a 3-dimensional tensor tree code C^3: Given a word $\mathbf{w} \in \mathbb{N}^{n_1 \times n_2 \times n_3}$ as input,

(1) Sample an axis $d \in [3]$.

(2) Sample a two-dimensional plane pl that is orthogonal to axis d, where the plane that is at location i on d is sampled with probability $\sigma_{n_d}(i)$.

(3) Accept if and only if the restriction of $\mathbf{w}$ to pl is in C^2.

Using the insight that tree weight can be "measured" by the suffix distribution we are able to adapt the analysis of Viderman [Vid15] to obtain a local testability theorem for tensor tree codes. Specifically, we prove that r-fold tree tensors are locally testable with respect to *tree distance at all lines*, assuming the base code has such a distance guarantee. In fact, we show local testability for a strictly weaker notion of distance, namely *suffix distance*, which tightly matches our test; Suffix distance between two words $\mathbf{w}, \mathbf{w}' \in \Sigma^{n_1 \times n_2 \times n_3}$ is defined as the expected disagreement between the words at a point $i = (i_1, i_2, i_3)$ sampled randomly with probability

$$\sigma_{n_1 \times n_2 \times n_3} = \sigma_{n_1}(i_1) \cdot \sigma_{n_2}(i_2) \cdot \sigma_{n_3}(i_3). \tag{2}$$

In particular, the above is an extension of the suffix distribution to 3 dimensions, and can be extended to general r-dimensional tensors. A visualization of the two-dimensional suffix distribution as a "heatmap" is given in Figure 2.

2.4 Flattening Tensors

A standard tensor code that is an LTC immediately gives a (one-dimensional) block code: We can "flatten" any high-dimensional tensor $\mathbf{w}$ via any arbitrary embedding of of the high-dimensional space of coordinates to one dimension; Hamming distance is still preserved and there are no structural requirement over the encoding function (see Footnote 6).

We aim for a similar solution in the context of tree codes. In general, to "flatten" a tensor code C and obtain a regular tree code $\overline{C}$ in one dimension, we must specify a way to traverse over the high-dimensional space $\mathbb{N}^r$, virtually defining a 1-1 mapping between any integer $i \in \mathbb{N}$ and the i^{th} r-dimensional coordinate to be reached by the walk, which we denote by $\mathbf{i} \in \mathbb{N}^r$. Given such a traversal, to encode the i^{th} codeword symbol given the i^{th} message symbol, one simply computes $C_{\mathbf{i}}(\mathbf{m}_{\leq \mathbf{i}})$. A careful reader may already notice that, unlike in the standard setting, the flattening for tensor tree codes cannot be performed via an arbitrary traversal. Indeed, to obtain an online encoding function under this outline, our traversal over $\mathbb{N}^r$ must be *monotone*. That is, by the time the walk reaches coordinate $\mathbf{i} \in \mathbb{N}^r$, it must have visited all coordinates $\mathbf{i}' \leq \mathbf{i}$, $\mathbf{i}' \neq \mathbf{i}$.

Things get more complicated, however, when we think of the other two properties that we would like to preserve under the flattening, namely distance and local testability.

The Distance Problem. To begin with, tree distance is highly *order-sensitive* by the simple fact that permutations do not preserve it (as opposed to Hamming distance, for instance). Thus, different choices of traversal might give different distance between codewords. It is not even clear whether it is possible to turn tree distance at all lines in a high-dimensional tensor to tree distance in one-dimension. To see why this can be an issue, imagine, for instance, that the first disagreement between two 2-dimensional codewords occurs at location $\mathbf{i} = (i_1, i_2)$. It can be the case that all other disagreements occur in the 2-dimensional suffix after $\mathbf{i}$, namely at coordinates $\mathbf{i}' \geq \mathbf{i}$. Therefore, intuitively speaking, for the distance to be preserved to some extent, the traversal must "concentrate" in this 2-dimensional suffix right after it reaches the coordinate $\mathbf{i}$. This is virtually impossible since, on the one hand, the encoding procedure (and, therefore, the traversal) is independent in $\mathbf{i}$ and, on the other hand, a "symmetric" traversal that advances somewhat evenly throughout the space, will enter the suffix of errors too infrequently (roughly speaking, once every $\sqrt{n}$ steps, or $n^{(r-1)/r}$ in the general case).

Local Testability. Local testability does not convert smoothly either. Imagine a word w of length n. To test whether w is in the flattened code $\overline{C}$, it is natural to look at its "lifting" back to the high-dimensional space of coordinates. The lifting, which we denote by $\mathbf{w}$, is an r-dimensional object. However, for most values of n, $\mathbf{w}$ is *not* a "full" tensor, namely it does not populate a full r-dimensional rectangle, but rather only a *partial* tensor. It is not clear, then, how to test such an object since we are equipped only with a local test for tensors, which we cannot apply over $\mathbf{w}$ directly.

Another obvious gap is that the tester we have for C is with respect to tree distance at all lines (and, in particular, w.r.t suffix distance), while the tester we seek for $\overline{C}$ is w.r.t. tree distance over the one-dimensional space that accommodates the flattened tensor.

Despite its incompatibility, we can still think of a tester $\overline{T}$ for the flattened code $\overline{C}$ that utilizes a tester T for the tensor code C w.r.t. suffix distance. On input a word w, our tester applies T over maximal full sub-tensors inside the lifting $\mathbf{w}$: define the set of *endpoints* in $\mathbf{w}$ to be the set of all maximal coordinates in $\mathbf{w}$ (i.e. any $\mathbf{e}$ such that $\mathbf{w}$ does not populate coordinates $\mathbf{i} \geq \mathbf{e}$). Our tester $\overline{T}$, then, samples a random such endpoint $\mathbf{e}$ and applies T to the (full) tensor ending at $\mathbf{e}$.

Such a test makes sense on the intuitive level for the following reason. When w is far from the code $\overline{C}$ in tree distance, there exists a suffix in w where many errors occur. By the presumed monotonicity of our encoding function (see above), the embedding of such a suffix of errors in high dimensions must induce errors in the range of at least one of the endpoints which, in turn, results

in high suffix distance in the corresponding sub-tensor. Formalizing this intuition turns out to be highly non-trivial and requires careful technical analysis. This is because, potentially, the closest codeword to the restriction to any sub-tensor might not be consistent with the closest codeword to w. It might be the case, for example, that any sub-tensor is close enough to its closest codeword, yet none of these closest codewords are equal, and the word w as a whole is far from the code. We refer the reader to Section 6.2 for further elaboration and a full proof.

Even with the above argument being proven – that high tree distance of w translates to high suffix distance in one of the endpoints of $\mathbf{w}$ – the local testability of a flattened code is not yet confirmed. Notice that, since our tester T samples a uniformly random endpoint, it is crucial that the number of endpoints is small for soundness to be reasonable. Hence, even the local testability of $\overline{C}$, similarly to its distance, heavily relies on the design of our traversal (at least according to this outline).

With the above observations in mind, we proceed to describe the traversal which, as described above, immediately defines the flattened code $\overline{C}$.

The Traversal. Our traversal over $\mathbb{N}^3$ is a recursive procedure, which "hops" between different parts in $\mathbb{N}^3$ of lower dimensions, i.e. 2-dimensional rectangles, 1-dimensional segments, and 0-dimensional points, and traverses over them recursively. Traversing a one-dimensional segment is straight-forward given the traversal has to be monotone. We now describe our traversal over three dimensions. The traversal over two dimensions follows the same abstract logic. We complement our description with helpful visualizations in Figs. 3 and 4.

The traversal is performed in layers, where by the end of every layer we have traversed all coordinates in a slightly larger cube. In more details, the first layer consists simply of the coordinate (1,1,1). In the second layer, we traverse all additional points in the cube ending at (2,2,2). At the end of the third layer we reach (3,3,3), and so on. It remains, then, to determine the order among the coordinates in each such layer, keeping in mind that the order has to be monotone. At a high level, observe that each layer consists of the following parts that "wrap around" the cube from the previous layer: First, there are the three faces corresponding to each of the 3 directions in which we can "inflate" the cube further; at layer t these correspond to all coordinates that have t at one of the dimensions (and only one). Second, there are the pairwise intersections between the continuation of these faces, namely the three segments corresponding to coordinates where the value t is found in *two* out of the three dimensions. Lastly, there is the point where all the continuation of the three segments intersect, namely the furthest corner in the layer, (t, t, t). Given this partitioning, our strategy, as mentioned above, is to recursively traverse each of these objects (the faces, segments and corner), which are of lower dimensionality.

In the general case of r-dimensions, any layer consists of r $(r-1)$-dimensional "faces", between which there are $\binom{r}{2}$ "intersections" of dimension $r-2$ and, in general, $\binom{r}{d}$ subspaces of dimension $r-d$, each corresponding to inflating the cube from the previous layer in d different directions. Notice that such a traversal is monotone as long as we are careful to make these recursive calls in a descending order in their size, starting with the 2-dimensional faces and ending up with the "0-dimensional" corner.

A formal description of the traversal can be found in Fig. 6.

As hinted above, we can show that our traversal exhibits a small number of endpoints at every step. In fact, there are always at most $r \cdot 2^r$ endpoints. This allows us to carry out the above outline for local testability and obtain a locally testable flattened tensor. That is, a locally testable

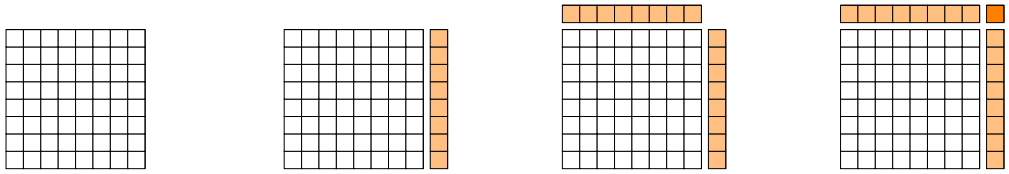

Figure 3: The ninth layer in the traversal over $\mathbb{N}^2$. In orange are the recursive calls to the traversal over one-dimensional lines and a zero-dimensional point, ordered from left to right. The choice to traverse the column before the row is arbitrary.

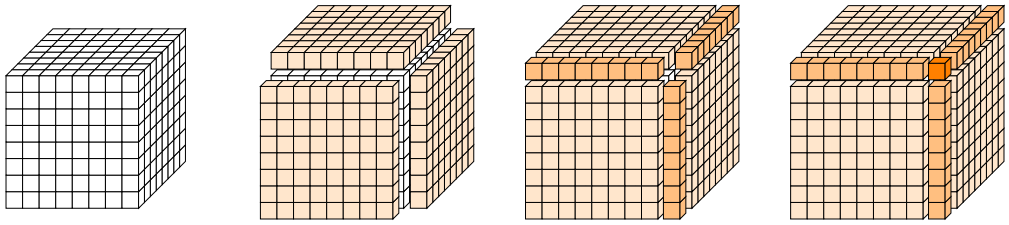

Figure 4: The ninth layer in the traversal over $\mathbb{N}^3$. In orange are the recursive calls to the traversal over segments of lower dimensionality, ordered from left to right. The order between calls to the different segments in each step of the illustration is determined arbitrarily.

tree code! We should not declare victory yet, however, since the distance issue remains: While our flattened code is a tree code, it does not guarantee tree distance between its codewords.

We proceed to present the last step in our work, where we bootstrap our flattened code to a tree code that actually satisfies tree distance, while preserving local testability. Since our bootstrapping uses randomness, we obtain a *probabilistic* tree code (see Definition 1.3).

Bootstrapping Suffix Distance to Tree Distance. At the heart of our transformation is a simple idea for generically bootstrapping suffix distance to tree distance in the probabilistic setting. Given that tree distance is a stronger notion than suffix distance in the deterministic setting, such a bootstrapping implies that the notions are in fact equivalent in the probabilistic setting.

Assume we have a tree code C_{S} that satisfies suffix distance. For simplicity, let us assume it is a one-dimensional tree code. Recall the probabilistic definition of tree code, where the goal is to create large tree distance between the encodings of two distinct messages m and m' that are chosen interactively by a channel $\mathcal{C}$ as depicted in Fig. 1.

Our high level idea is to copy, when encoding the i^{th} symbol of the message, randomly chosen codeword symbols from the past, hoping to "catch" a coordinate where a disagreement between $C_{\mathsf{S}}(m)$ and $C_{\mathsf{S}}(m')$ has already occurred. Assuming m and m' already exhibit a disagreement before time i, it holds by the assumed distance of C_{S} that the encodings $c = C_{\mathsf{S}}(m_{<i})$ and $c' = C_{\mathsf{S}}(m_{<i'})$ are *far* in suffix distance. What this means is that c and c' are expected to disagree, with good probability, over a random point i', sampled from the suffix distribution σ_{i-1} (defined in Eq. (1)). Consequently, we sample sufficiently many such previous coordinates $i' < i$, according to the suffix distribution, and copy $(i', c_{i'})$ to the i^{th} symbol. Notice that, since the randomness used at coordinate i is sampled after $m_{<i}$ and $m_{<i'}$ are fixed, an error will be caught with high probability, regardless of the choice of $\mathcal{C}$ for the messages $m_{<i}$ and $m'_{<i}$.

Overall, our probabilistic code, which we denote by C_{T}, consists of three parallel *tapes*: (i) An

encoding tape where the message is encoded under C_{S}, (ii) a *sample tape* that, at coordinate i, contains a polylogarithmic number of samples from previous coordinates in the encoding tape, and (iii) a *randomness tape* where the random bits used for the sampling are registered.

The argument for tree distance of C_{T} is as follows. Let m, m' and ρ be the two messages and randomness obtained from the interaction with $\mathcal{C}$ after n rounds. Following the above reasoning, we prove that $c = C(m;\rho)$ and $c' = C(m';\rho)$ must be far; again, by the way ρ is sampled, it must "catch" disagreements between apriori fixed codeword symbols. We argue that this already implies that c is far from *any* encoding of m' under any randomness ρ'. To see this, observe that in any attempt to "fix" a symbol in c' to match the value in c is doomed to fail; in particular, to fix a pair $(i', c'_{i'})$ in the sample tape, one must change the value of $c'_{i'}$ (i' is equal in both codewords since they use the same randomness). This is impossible to do unless one changes the value of i', in which case the codewords remain in disagreement.

We stress that, fortunately, the transformation from C_{S} to C_{T} preserves local testability. A tester for C_{T} applies the tester for C_{S}, over the encoding tape, and performs consistency checks of the sample tape with the encoding and randomness. The consistency checks consist of standard sub-sampling, yet, not surprisingly, must use the suffix distribution. For more details, we refer the reader to the description of the test in Figure 7.

Collecting Errors from All Endpoints. Lastly, we generalize the above transformation to high-dimensional tree codes. We generalize the above in a most straight-forward way: Suffix distribution is replaced with its high-dimensional extension (see Eq. (2)) and, more importantly, at coordinate $\mathbf{i} \in \mathbb{N}^r$, we collect samples from *any* maximal sub-tensor, namely from the range of any endpoint (in the one-dimensional case there is only one endpoint – the last coordinate i). Tree distance follows since, if m and m' diverge, they must disagree at least in one of the maximal sub-tensors, in over which their encodings must exhibit large suffix distance (by the assumption on C_{S}). Further, since the number of endpoints is bound by constant, the transformation still introduces only a polylogarithmic blow-up in the rate of the code.

3 Preliminaries

Basic Notation. For a string $w \in \Sigma^n$, we use $w_{\leq i}$ to denote its prefix of length i and by $w_{\geq i}$ its suffix starting at position i (of length $n - i + 1$). For $Q \subseteq [n]$, we use w_Q to denote the projection of the string to coordinates in Q. We denote by $\mathbf{0}$ the all-zero tensor (where size is always clear from context), by $\mathbf{1}_j$ the j^{th} unit vector, and by $\mathbf{1} = \sum_j \mathbf{1}_j$ the all-ones tensor.

High-dimensional Notation. We typically use boldface to denote high-dimensional objects, e.g. $\mathbf{w}$. We use an r-tuple of integers $N = (n_1, \ldots, n_r) \in \mathbb{N}^r$ to denote the *size* of a rectangular r-dimensional tensor that has length n_d at dimension $d = 1, \ldots, r$. We sometimes write $N = n_1 \times \cdots \times n_r$ to implicitly specify that N is such a size parameter, and denote the r-dimensional space by $\Sigma^N = \Sigma^{n_1 \times \cdots \times n_r}$. For any high-dimensional size parameters $N_1 \in \mathbb{N}^{r_1}$ and $N_2 \in \mathbb{N}^{r_2}$, we let $N_1 \times N_2$ denote their concatenation, namely the size of the corresponding $(r_1 + r_2)$-dimensional product space. We use the notation $[N] = [n_1] \times \cdots \times [n_r]$ to denote the *range* of coordinates of a tensor of size N and, consequently, $|[N]| = \prod_i n_i$. We write $\mathbf{w} \in \Sigma^I$, for a subset $I \subseteq \mathbb{N}^r$ (that is not necessarily a rectangle), to refer to a *partial* tensor that is defined only over the coordinates in I (observe that such a partial tensor can be viewed as a function $\mathbf{w} : I \to \Sigma$).

We additionally use boldface to denote coordinates in high-dimensional spaces. For any two such coordinates $\mathbf{i}, \mathbf{j} \in [N]$, letting $\mathbf{i} = (i_1, \ldots, i_r)$ and $\mathbf{j} = (j_1, \ldots, j_r)$, we say that $\mathbf{i} < \mathbf{j}$, or $\mathbf{i} \leq \mathbf{j}$, if and only if $i_k < j_k$ or, resp., $i_k \leq j_k$, for all $k = 1, \ldots, r$. Note that this induces a partial order relation over the space of coordinates. In particular, we use $\not<$ and, respectively, $\not\leq$ to denote the complements of the aforementioned relations (these are not equivalent to $\geq$ or $>$).

3.1 Codes in High Dimension

Definition 3.1 (Monotone Set of Coordinates)**.** *We say that $I \subset \mathbb{N}^r$ is* monotone *if $\mathbf{i} \in I$ and $\mathbf{j} \leq \mathbf{i}$ implies $\mathbf{j} \in I$ for any $\mathbf{i}, \mathbf{j} \in \mathbb{N}^r$.*

Definition 3.2 (Code)**.** *Let $r \in \mathbb{N}$. An r-*dimensional code *C over an alphabet Σ is simply a subset $C \subseteq \bigcup_{N \in \mathbb{N}^r} \Sigma^N$. A* codeword *in C is any $w \in C$.*

We sometimes overload notation and use C to refer also to an injective encoding function that maps a message space to codewords in C (in cases where the encoding function will be clear from context).

For a set $Q \subseteq \mathbb{N}^r$, we use C_Q to denote the restriction of the code C to the coordinates Q, Namely, $C_Q = \{w_Q \mid w \in C\}$.

Definition 3.3 (Code Distance)**.** *Let $r \in \mathbb{N}$. Let $\Delta = \{\Delta_N : \Sigma^N \times \Sigma^N \to [0,1]\}_{N \in \mathbb{N}^r}$ be a distance metric. For any $N \in \mathbb{N}^r$ such that $C \cap \Sigma^N \neq \emptyset$, we denote*

$$\Delta_N(C) = \min_{\substack{\mathbf{w}, \mathbf{w}' \in C \cap \Sigma^N \\ \mathbf{w} \neq \mathbf{w}'}} \Delta(\mathbf{w}, \mathbf{w}'),$$

and say that an (r-dimensional) code C has Δ-distance *$\delta : \mathbb{N}^r \to [0,1]$ if $\Delta_N(C) \geq \delta(N)$ for all such $N \in \mathbb{N}^r$.*

We define the distance between any $\mathbf{w} \in \Sigma^N$ and a code C (such that $C \cap \Sigma^N \neq \emptyset$) as $\Delta(\mathbf{w}, C) = \min_{\mathbf{w}' \in C \cap \Sigma^N} \Delta(\mathbf{w}, \mathbf{w}')$.

3.2 Local Testing of Codes

In the task of *locally testing* a code C, a *tester* is a randomized algorithm that is given oracle access to a word $w \in \Sigma^N$, and its goal is to correctly determine whether w is a codeword in C with good probability, by looking only at a bounded number of locations in w.

In this work, and following standard literature,[8] we will consider the special case of testers that make non-adaptive queries to w and have perfect completeness (i.e. accept a codeword $w \in C$ with probability 1). Without loss of generality, one may assume that the tester computes a (randomized) query set $Q \subseteq [N]$, queries w at the locations specified by Q to obtain w_Q, then outputs 1 if and only if $w_Q \in C_Q$. Note that such a canonical tester has perfect completeness, and soundness error may be calculated as $\Pr[w_Q \notin C_Q]$.

Thus, it is possible to reduce the tester to the randomized algorithm that outputs the query set Q, as we do in the definition below. Further, since our general construction relies on composing a sequence of testers, we define stronger notions of local testing.

[8] We generalize standard LTC notions to fit in our high-dimensional perspective.

Definition 3.4 (Locally Testable Codes)**.** *Let $r \in \mathbb{N}$ and $q : \mathbb{N}^r \to \mathbb{N}$. A* q-query tester *(for r-dimensional codes) is an ensemble $T = \{T_N\}_{N \in \mathbb{N}^r}$ where, for any $N \in \mathbb{N}^r$, T_N is a distribution over subsets $Q \subseteq [N]$ of size $|Q| \leq q(N)$.*

- *For $q : \mathbb{N}^r \to \mathbb{N}$, $\epsilon, \delta : \mathbb{N}^r \to [0,1]$, we say that an r-dimensional code C is a* $(\Delta, q, \epsilon, \delta)$-locally testable code *(or $(\Delta, q, \epsilon, \delta)$-LTC for short) if there exists a q-query tester such that, for any $N \in \mathbb{N}^r$ and $w \in \Sigma^N$, if $\Delta(w, C) \geq \delta(N)$ then $\Pr_{Q \leftarrow T_N}[w_Q \notin C_Q] \geq \epsilon(N)$.*
- *For $q : \mathbb{N}^r \to \mathbb{N}$, $\epsilon : \mathbb{N}^r \to [0,1]$, we say that C is a* (Δ, q, ϵ)-strong LTC *if there exists a q-query tester such that, for any $N \in \mathbb{N}^r$ and $w \in \Sigma^N$ it holds that $\Pr_{Q \leftarrow T_N}[w_Q \notin C_Q] \geq \epsilon(N) \cdot \Delta(w, C)$.*
- *For $q : \mathbb{N}^r \to \mathbb{N}$, $\epsilon, \rho : \mathbb{N}^r \to [0,1]$, we say that C is a* (Δ, q, ρ)-robust LTC *if there exists a q-query tester such that, for any $N \in \mathbb{N}^r$ and $w \in \Sigma^N$ it holds that $\mathbb{E}_{Q \leftarrow T_N}[\Delta(w_Q, C_Q)] \geq \rho(N) \cdot \Delta(w, C)$.*

We say that C is simply (Δ, q, δ)-LTC if there exists a constant $\epsilon > 0$ such that C is $(\Delta, q, \delta, \epsilon)$-LTC.

The following proposition follows from straight-forward amplification.

Proposition 3.5. *If C is a $(\Delta, q, \epsilon, \delta)$-LTC then it is $(\Delta, s \cdot q, 1 - e^{-\epsilon \cdot s}, \delta)$-LTC for any $s \in \mathbb{N}$.*

3.3 Tensoring Block Codes

Definition 3.6 (Block Code)**.** *A (one-dimensional)* block code *C over Σ with* block-length *$n \in \mathbb{N}$ is a subset of codewords of fixed length n, namely a code $C \subseteq \Sigma^n$.*

More generally, an r-dimensional block code *C* with block-length *$N \in \mathbb{N}^r$ is a subset $C \subseteq \Sigma^N$.*

Definition 3.7 (Linear Block Code)**.** *Let $\mathbb{F}$ be a field and $r \in \mathbb{N}$. We say that an r-dimensional block code C over $\mathbb{F}$ is a* linear (block) code *if the linear combination of any two codewords in C is also a codeword in C. In particular, C is a vector space over $\mathbb{F}$.*

We say that such a linear code C is an $[N, k]_{\mathbb{F}}$-code, for $N \in \mathbb{N}^r$ and $k \in \mathbb{N}$, if it has block-length N and $\dim_{\mathbb{F}} C = k$.

We say that a linear transformation $\mathbf{C} : \mathbb{F}^K \to \mathbb{F}^N$ is a generator *for an $[N, k]_{\mathbb{F}}$-code C if and only if $|[K]| = k$ and $C = \{\mathbf{C}(\mathbf{m}) \mid \mathbf{m} \in \mathbb{F}^K\}$.*

Definition 3.8 (Tensor Product of Block Codes)**.** *Let C_1 and C_2 be (possibly high-dimensional) block codes over Σ^{N_1} and, respectively, Σ^{N_2}. The tensor product of C_1 and C_2, denoted by $C_1 \otimes C_2$, consists of all codewords $\mathbf{w} \in \Sigma^{N_1 \times N_2}$, such that any column*[9] *in $\mathbf{w}$ is a codeword in C_1 and any row in $\mathbf{w}$ is a codeword in C_2.*

Further, we define $C^1 = C$ and $C^r = C^{r-1} \otimes C$ for any $r > 1$.

Proposition 3.9. *If $C_1 \subset \mathbb{F}^{N_1}, C_2 \subset \mathbb{F}^{N_2}$ are linear codes of dimensions r_1 and, resp., r_2, then it holds that their tensor product code (see Definition 3.8) is*

$$C_1 \otimes C_2 = \mathbf{Span}(\{\mathbf{c}_1 \otimes \mathbf{c}_2\}_{\substack{\mathbf{c}_1 \in C_1 \\ \mathbf{c}_2 \in C_2}}),$$

[9] In the general case with more than two dimensions, "columns" of $\mathbf{w}$ are elements in Σ^{N_1} and are indexed by N_2-coordinates, while "rows" are in Σ^{N_2} and have N_1-coordinates.

where $\mathbf{c}_1 \otimes \mathbf{c}_2 \in \mathbb{F}^{N_1 \times N_2}$ *is the* $(r_1 + r_2)$*-dimensional tensor where the* $\mathbf{i}^{th}$ *entry, for* $\mathbf{i} = (\mathbf{i}_1, \mathbf{i}_2) \in [N_1] \times [N_2]$*, is equal to* $\mathbf{c}_1[\mathbf{i}_1] \cdot \mathbf{c}_2[\mathbf{i}_2]$*.*

Further, if $\mathbf{C}_1 : \mathbb{F}^{K_1} \to \mathbb{F}^{N_1}$ *and* $\mathbf{C}_2 : \mathbb{F}^{K_2} \to \mathbb{F}^{N_2}$ *are generators of* C_1 *and, resp.,* C_2*, then it holds that* $\mathbf{C}_1 \otimes \mathbf{C}_2 : w \mapsto \mathbf{C}_1 w \mathbf{C}_2^T$ *is a generator for* $C_1 \otimes C_2$*, where* $\mathbf{C}_2^T$ *is the linear map* $f \mapsto f \circ \mathbf{C}_2$*.*

Definition 3.10 (Plane Tester)**.** *Let* $r \in \mathbb{N}$ *and* $N = (n_1, \ldots, n_r) \in \mathbb{N}^r$*. We denote by* $PL(N)$ *the set of all* $(r-1)$*-dimensional subspaces in* $[N]$*. Namely,*

$$PL(N) = \{pl_{(d,i)} : d \in [r], i \in [n_d]\},$$
$$\text{where} \quad pl_{(d,i)} = \big\{(i_1, \ldots, i_{d-1}, i, i_{d+1}, \ldots, i_r) : i_{d'} \in [n_{d'}], \quad \forall d' \neq d\big\}.$$

When N *is clear from context, we use* PL *as a shorthand for* $PL(N)$*.*

A plane tester *for an* r*-dimensional code is a tester* $T = \{T_N\}_{N \in \mathbb{N}^r}$ *(see Definition 3.4)*[10] *where, for any* $N \in \mathbb{N}^r$*,* T_N *always outputs an* $(r-1)$*-dimensional plane* $pl \in PL(N)$*.*

Note that plane tester merely needs to specify a distribution over d (which in our case will be uniform) and over i (which will not be uniform, but will be independent of d).

Proposition 3.11. *Let* $\mathbb{F}$ *be a field and* $\Delta = \{\Delta_N : \mathbb{F}^N \to [0,1]\}_{N \in \mathbb{N}^*}$ *be any distance metric (defined over any number of dimensions). Let* C *be a block code and let* C^r *be its* r*-fold tensor (see Definition 3.8). Assume* C^r *is a* $(\Delta, n^{r-1}, \alpha_r)$*-robust LTC (see Definition 3.4) via a plane tester (see Definition 3.10) for any* $r \geq r_0$*, where* $n = L_\infty(N)$*. Then it is also a* $(\Delta, n^{r_0-1}, \prod_{d=r_0}^{r} \alpha_d)$*-robust LTC and, in particular, a* $(\Delta, n^{r_0-1}, \prod_{d=r_0}^{r} \alpha_d)$*-strong LTC.*

Proof. For any $d \geq r_0$, let $T^{(d)}$ be the plane tester for C^d that makes it a $(\Delta, n^{d-1}, \alpha_d)$-robust LTC. Given $r > 2$, consider the tester T^* which at the beginning invokes $T^{(r)}$ to obtain an $(r-1)$-dimensional plane $Q(r-1) \in PL(N)$ then proceed recursively; At iteration $i = 1, \ldots, r - r_0$, it invokes tester $T^{(r-i)}$ on the view $Q(r-i)$ sampled by the previous tester (recall this is a an $(r-i)$-dimensional plane and, therefore, corresponds to an underlying candidate codeword in C^{r-i}) to obtain a view $Q(r-i-1)$ with one dimension less. The tester T^* eventually outputs a query set of size n^{r_0-1}; this is the (r_0-1)-dimensional plane $Q(r_0-1)$ sampled at the last step of the recursion by $T^{(r_0)}$. To analyze the robustness of T^*, fix $\mathbf{w} \in \mathbb{F}^N$. Then, it holds

$$\begin{aligned}
\mathbb{E}_{Q(r_0-1)}[\Delta(\mathbf{w}[Q(r_0-1)], C^{r_0-1})] &\geq \alpha_{r_0} \cdot \mathbb{E}_{Q(r_0)}[\Delta(\mathbf{w}[Q(r_0-1)], C^{r_0})] \\
&\geq \alpha_{r_0} \cdot \alpha_{r_0+1} \cdot \mathbb{E}_{Q(r_0+1)}[\Delta(\mathbf{w}[Q(r_0-1)], C^{r_0+1})] \\
&\geq \ldots \\
&\geq \prod_{d=r_0}^{r} \alpha_d \cdot \Delta(\mathbf{w}, C^r),
\end{aligned}$$

where $Q(r_0-1)$, $Q(r_0)$, $\ldots$ are as sampled by T^*. Further,

$$\begin{aligned}
\Pr_{Q(r_0-1)}[\mathbf{w}[Q(r_0-1)] \notin C^{r_0-1}] &= \Pr[\Delta(\mathbf{w}[Q(r_0-1)], C^{r_0-1}) > 0] \\
&\geq \mathbb{E}_{Q(r_0-1)}[\Delta(\mathbf{w}[Q(r_0-1)], C^{r_0-1})]
\end{aligned}$$

(the last inequality is by the fact that $\Delta(\cdot,\cdot)$ has maximal value 1) and, therefore, we get the strong LTC property as well. □

[10] Although we define testers for one-dimensional codes, the generalization to high-dimensional codes is straightforward.

4 Tensor Tree Codes

In this section, we define a tensor product of tree codes and lay down the formal framework that enables us to leverage its features. The framework rests on two main components: We define a high-dimensional analog of tree codes through which we identify a notion of *online encoding in high dimensions* that exists in tensor tree codes. Additionally, we propose a meaningful adaptation of tree distance to the high-dimensional setting that is achievable by a tensor product and is useful to our purpose (as will be shown in later sections).

4.1 High-Dimensional Tree Codes

An *r-dimensional tree code* takes as input an r-dimensional message $\mathbf{m} \in \Sigma_{\mathsf{in}}^N$, of some r-dimensional size $N \in \mathbb{N}^r$, and outputs a codeword $\mathbf{w} \in \Sigma_{\mathsf{out}}^N$. This new notion is a generalization of (one-dimensional) tree codes in the sense that its encoding can be done in an online fashion; such a code is defined by an ensemble of functions $\{C_N\}_{N \in \mathbb{N}^r}$ where, on input $\mathbf{m} \in \Sigma^N$, the function C_N outputs the "last" symbol in the (high-dimensional) codeword $\mathbf{w} = C(\mathbf{m}) \in \Sigma^N$. Here, we naturally think of the last symbol to be the one at the coordinate taking maximal values in all dimensions, e.g. the corner at (n_1, n_2) in a rectangle of size $n_1 \times n_2$. This is consistent with the partial order over the space $\mathbb{N}^r$ that we have defined in Section 3.

Definition 4.1 (High-Dimensional Tree Code)**.** *Let $r \in \mathbb{N}$. An* r-dimensional tree code *is a function ensemble $C = \{C_N : \Sigma_{\mathsf{in}}^N \to \Sigma_{\mathsf{out}}\}_{N \in \mathbb{N}^r}$ that defines an* encoding function *which on input $\mathbf{m} \in \Sigma_{\mathsf{in}}^N$ outputs*

$$\mathbf{w} = C(\mathbf{m}) \in \Sigma_{\mathsf{out}}^N \qquad \textit{where} \quad \mathbf{w_i} = C_{\mathbf{i}}(\mathbf{m}_{\leq \mathbf{i}}), \quad \textit{for every } i \in [N].$$

4.2 The Tensor Product of Tree Codes

We define tensor products for a special case of tree codes, namely *linear vector tree codes*. Linear vector tree codes are linear tree codes where the output alphabet is not any arbitrary ring, but rather is a vector space of the input alphabet (which, consequently, must be a field). Such codes have the special property that a vector of codeword symbols can be parsed as an input message to the encoding function. This property is what essentially allows us to apply a tensor product over such codes and obtain a well-defined encoding function satisfying the notion required in Definition 4.1.

In the definition below, we extend the notion of linear codes to high dimensions and define the special case of linear vector tree codes.

Definition 4.2 (Linear Vector Tree Codes)**.** *We say that an r-dimensional tree code $C = \{C_N : \Sigma_{\mathsf{in}}^N \to \Sigma_{\mathsf{out}}\}_{N \in \mathbb{N}^r}$ is* linear *if Σ_{in} and Σ_{out} are both rings and, for every $N \in \mathbb{N}^r$, C_N is a linear function (and, therefore, so is the encoding function C), i.e. $C_N(u + v) = C_N(u) + C_N(v)$ and $C_N(-u) = -C_N(u)$ for any $u, v \in \Sigma_{\mathsf{in}}^N$.*

Let $\mathbb{F}$ be a field and let $\ell : \mathbb{N}^r \to \mathbb{N}$. We say that $C = \{C_N : \Sigma_{\mathsf{in}}^N \to \Sigma_{\mathsf{out}}\}_{N \in \mathbb{N}^r}$ is a linear vector (r-dimensional) tree code over $\mathbb{F}$ with rate $1/\ell$ *if it is a linear tree code with $\Sigma_{\mathsf{in}} = \mathbb{F}$ and, for every $N \in \mathbb{N}^r$, the alphabet for N^{th} codeword symbols comes from a vector space of dimension $\ell(N)$ over $\mathbb{F}$ (namely, $C_N : \mathbb{F} \to \mathbb{F}^L$ for some $L \in \mathbb{N}^r$ such that $|[L]| = \ell(N)$).*

In general, one can think of a tree code C as an infinite collection of block codes, each corresponding to a different input length; For any length $n \in \mathbb{N}$, we can look at the block code of all

codewords in C that are of length n, i.e $C \cap \Sigma_{\mathsf{out}}^n$. In particular, it holds that a linear vector tree code can be viewed as a collection of *linear* block codes.

Remark 4.3. *Let $\ell : \mathbb{N} \to \mathbb{N}$ and let C be a one-dimensional rate-$(1/\ell)$ linear vector tree code over a field $\mathbb{F}$. Then, for any $n \in \mathbb{N}$, letting $n' = \sum_{i=1}^{n} \ell(i)$, the restriction of C to to messages of length n is a linear $[n', n]_{\mathbb{F}}$-code with a lower-triangular generator matrix $\mathbf{C}_n \in \mathbb{F}^{n' \times n}$ where*

$$\mathbf{C}_n = \begin{pmatrix} C_1 & 0^{\ell(1)\times(n-1)} \\ C_2 & 0^{\ell(2)\times(n-2)} \\ \vdots & \vdots \\ C_n & \end{pmatrix}.$$

More generally, for an r-dimensional rate-$(1/\ell)$ linear vector tree code $C = \{C_N : \mathbb{F} \to \mathbb{F}^{\ell_1(N)\times\cdots\times\ell_r(N)}\}$,[11] the restriction of C to messages of length $N = n_1 \times \cdots \times n_r$ is a linear $[N', |[N]|]_{\mathbb{F}}$-code with a generator $\mathbf{C}_N : \mathbb{F}^N \to \mathbb{F}^{N'}$, where $N' = n'_1 \times \cdots \times n'_r$ for $n'_d = \sum_{i=1}^{n_d} \ell_d(i)$, and

$$\mathbf{C}_N : \mathbf{m} \mapsto (C_{\mathbf{i}}(\mathbf{m}_{\leq \mathbf{i}}))_{\mathbf{i} \in [N]}.$$

The above gives a characterization of linear vector tree codes as an infinite collection of linear block codes of increasing block-length.

We are now prepared to define the tensor product of linear vector codes. Our definition is straight-forward given the definition of tensor product for (linear) block codes (given in Definition 3.8 and Proposition 3.9) and the analogy made in Remark 4.3 between linear vector tree codes and infinite collections of linear block codes. The idea is very simple: The tensor product of one-dimensional tree codes C_1 and C_2 is defined by the online function (online in the "high-dimensional sense" as in Definition 4.1) that, to output the N^{th} codeword symbol given an input message $\mathbf{m}$ of length $N = n_1 \times n_2$, applies the tensor product of the linear transformation corresponding to the n_1^{th} encoding function from C_1 with the n_2^{th} encoding function from C_2.

Note that, to allow for recursive tensoring, we define tensor product for tree codes that, possibly, are already high-dimensional.

Definition 4.4 (Tensor Product of Tree Codes)**.** *Let $C_1 = \{C_{1,N_1} : \mathbb{F}^{N_1} \to \mathbb{F}^{L_1(N_1)}\}_{N \in \mathbb{N}^{r_1}}$ and $C_2 = \{C_{2,N_2} : \mathbb{F}^{N_2} \to \mathbb{F}^{L_2(N_2)}\}_{N \in \mathbb{N}^{r_2}}$ be r_1- and, resp., r_2-dimensional linear vector tree codes (see Definition 4.2). We define the* tensor product *of C_1 with C_2, denoted by*

$$C_1 \otimes C_2 = \left\{(C_1 \otimes C_2)_N : \mathbb{F}^N \to \mathbb{F}^{L(N)}\right\}_{N \in \mathbb{N}^{r_1+r_2}}$$

as the $(r_1 + r_2)$-dimensional tree code where, for every $N = (N_1, N_2) \in \mathbb{N}^{r_1} \times \mathbb{N}^{r_2}$, we define $L(N) = L_1(N_1) \times L_2(N_2)$, and

$$(C_1 \otimes C_2)_N = C_{1,N_1} \otimes C_{2,N_2}$$

(see Proposition 3.9 for definition of $\otimes$ over linear maps).

Letting $C^1 = C$, we denote $C^r = C^{r-1} \otimes C$ for any $r > 1$.

[11] It holds $\ell(N) = \prod_d \ell_d(N)$ for all N.

In the following proposition, we confirm that the tensor product operation over tree codes essentially corresponds to applying the tensor product over their associated collections of block codes. This gives us a strong characterization of tensor tree codes as collections of tensor block codes. This is useful since the structure of tensors for block codes and their features are well understood.

Proposition 4.5. *Let $C_1 = \{C_{1,N_1} : \mathbb{F}^N \to \mathbb{F}^{L_1(N_1)}\}_{N\in\mathbb{N}^{r_1}}$ and $C_2 = \{C_{2,N_2} : \mathbb{F}^N \to \mathbb{F}^{L_2(N_2)}\}_{N\in\mathbb{N}^{r_2}}$ be linear vector tree codes. Let $\{\mathbf{C}_{1,N_1}\}_{N_1\in\mathbb{N}^{r_1}}$ and $\{\mathbf{C}_{2,N_2}\}_{N_2\in\mathbb{N}^{r_2}}$ be the infinite collections of linear block codes that correspond to the codes C_1 and C_2 due to Remark 4.3. Then, it holds that*

$$C_1 \otimes C_2 = \bigcup_{\substack{N_1\in\mathbb{N}^{r_1}\\ N_2\in\mathbb{N}^{r_2}}} \mathbf{C}_{1,N_1} \otimes \mathbf{C}_{2,N_2}.$$

Proof. To see why the equivalence holds, let us fix N_1 and N_2 and show $\mathbf{C}_{1,N_1} \otimes \mathbf{C}_{2,N_2}$ is the set of all codewords in $C_1 \otimes C_2$ corresponding to messages of size $N = N_1 \times N_2$. Observe that, since $\mathbb{F}^N$ is spanned by $\{\mathbf{m} \otimes \mathbf{m}'\}$ for all $\mathbf{m} \in \mathbb{F}^{N_1}$ and $\mathbf{m}' \in \mathbb{F}^{N_2}$, it holds that N-size codewords in $(C_1 \otimes C_2)$ are spanned by $\{(C_1 \otimes C_2)(\mathbf{m} \otimes \mathbf{m}') = C_1(\mathbf{m}) \otimes C_2(\mathbf{m}')\}$. By Proposition 3.9, this is exactly $\mathbf{C}_{1,N_1} \otimes \mathbf{C}_{2,N_2}$. □

4.3 Distance of The Tensor Code

So far we have defined a notion of high-dimensional tree codes that, in particular, captures the "online-ness" of the encoding function of a tensor tree code. We proceed to define a notion of tree distance that captures the distance guarantee that we obtain from our tensors. Notice that, by our definition, online encoding in high dimensions is substantially different from the standard online notion. Intuitively speaking, this online encoding is with respect to a *high-dimensional notion of time*, that consists of a multitude of intersecting lines, each defining a distinct "timeline".

Looking back at our tensor code and its structural properties (see Definition 3.8), we observe that it inherits the tree distance of the base code over any such timeline! More formally, any restriction of the code C^r over a line in the space satisfies the tree distance of C.

Definition 4.6 (Distance at Any Line)**.** *Let $r \in \mathbb{N}$ and let C be a high-dimensional code. Let Δ be any one-dimensional distance metric. We say that C has Δ-*distance δ at any line *if, for any distinct $\mathbf{w}, \mathbf{w}' \in C$, it holds, that*

$$\min_{\ell\in L:\mathbf{w}_\ell\neq\mathbf{w}'_\ell} \Delta(\mathbf{w}_\ell, \mathbf{w}'_\ell) \geq \delta(|\ell|),$$

where L is the set of lines in $[N]$, i.e. contains any subset of coordinates $\ell = \{(i_1, \ldots, i_{d-1}, i, i_{d+1}, \ldots, i_r) \mid i \in [n_d]\}$ for $d \in [r]$ and $i_{d'} \in [n_{d'}]$ for $d' \neq d$.

Remark 4.7. *For any metric Δ, any r-dimensional code C, and any subset of lines $L' \subseteq L$ (possibly forming a hyperplane), it holds that if C has Δ-distance at any line, then so does $C_{L'}$.*

We finish this section with the following corollary that is immediately implied by Definition 3.8.

Corollary 4.8. *Let $r \in \mathbb{N}$ and let C be an r-dimensional tree code with tree distance δ. Then, the tensor product C^r has tree distance δ at any line.*

5 Testing Tensor Tree Codes

In this section, we show that, similar to the case of standard tensor codes, tensor products of tree codes possess powerful local testability properties due to their combinatorial structure. We do so by generalizing the proof of Viderman [Vid15], who showed that the r-fold tensor product of *any* (standard) code, for $r \geq 3$, are locally testable w.r.t. Hamming distance. Specifically, we show that tensors of (possibly high-dimensional) tree codes, which themselves are high-dimensional tree codes as shown in the previous section, are locally testable w.r.t distance at any line. In fact, we show that they are locally testable w.r.t. a weaker notion of distance, which we call *suffix distance*.

5.1 Suffix Distance

We define suffix distance between two words of length n via a distribution σ over the locations 1 to n. Roughly speaking, σ samples location i with probability inverse-proportional to its distance from the end, i.e. proportional to $1/(n-i)$. Intuitively, this captures the impact a location i has on the tree distance between these two words. Suffix distance is then defined to be the expected disagreement between the two words for a symbol at location sampled from σ. Since we consider notions of tree distance in high dimensions as well, we also generalize suffix distance for high-dimensional tensors of length $N = n_1 \times \cdots \times n_r$ in the natural way; we define it to correspond to the distribution obtained by the product $\sigma_N = \sigma_{n_1} \otimes \cdots \otimes \sigma_{n_r}$.

Definition 5.1 (Suffix Distance)**.** *Let Σ be an alphabet. Define $H_n = \sum_{j=1}^{n} 1/j$ for any $n \in \mathbb{N}$ and $H_N = \prod_{n \in N} H_n$ for any $N \in \mathbb{N}^r$. For any $r \in \mathbb{N}$, $N = (n_1, \ldots, n_r) \in \mathbb{N}^r$ and coordinate $\mathbf{i} = (i_1, \ldots, i_r) \in [N]$, we define*

$$\sigma_N(\mathbf{i}) = \frac{1}{H_N} \cdot \prod_{j=1}^{r} \frac{1}{n_j - i_j + 1}.$$

Notice that $\sigma_N = \bigotimes_{j=1}^{m} \sigma_{n_j}$ and therefore σ_N is a probability density function. We sometimes override notation and use σ_N to denote the corresponding distribution over $[N]$.

Let $\mathbf{w}, \mathbf{w}' \in \Sigma^N$. We define the suffix distance *between $\mathbf{w}$ and $\mathbf{w}'$ as*

$$\Delta_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') = \Pr_{\mathbf{i} \leftarrow \sigma_N}[\mathbf{w}_{\mathbf{i}} \neq \mathbf{w}'_{\mathbf{i}}].$$

We further define $\omega_{\mathsf{S}}(\mathbf{w}) = \Delta_{\mathsf{S}}(\mathbf{w}, 0)$ to be the suffix weight *of $\mathbf{w}$.*

In the following lemma, we give a lower bound on the suffix distance between any two (possibly high-dimensional) words by the hamming distance in any of its (resp., high-dimensional) suffixes. This lower bound is crucial to establish the connection between suffix distance and tree distance at all lines (and immediately implies it in the one-dimensional case).

Lemma 5.2. *Let $r \in \mathbb{N}$ and $N \in \mathbb{N}^r$. Let $\mathbf{w}, \mathbf{w}' \in \Sigma^N$ and assume there exists $\mathbf{i}^* \in [N]$ such that $\Delta_{\mathsf{H}}(\mathbf{w}_{\geq \mathbf{i}^*}, \mathbf{w}'_{\geq \mathbf{i}^*}) \geq \delta$. Then, it holds that*[12]

$$\Delta_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') \geq \frac{1}{H_N} \cdot \delta - o(1),$$

where H_N is as defined in Definition 5.1.

[12] Asymptotic notation is with respect to N, specifically $L_{-\infty}(N)$.

Proof. Note that it is sufficient to give a lower bound on the suffix distance between $\mathbf{w}_{\geq \mathbf{i}^*}$ and $\mathbf{w}'_{\geq \mathbf{i}^*}$ since, by definition, it holds that

$$\Delta_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') \geq \frac{H_{N-\mathbf{i}^*+\mathbf{1}}}{H_N} \cdot \Delta_{\mathsf{S}}(\mathbf{w}_{\geq \mathbf{i}^*}, \mathbf{w}'_{\geq \mathbf{i}^*}). \tag{3}$$

We start by proving the inequality for one dimension. In fact, we prove the following more general statement. Let $x_1, \ldots, x_n$ be such that $0 \leq x_i \leq 1$ for all $i \in [n]$. We argue that

$$\mathbb{E}_{i \leftarrow \sigma_n}[x_i] \geq \frac{1}{H_n} \cdot \big(\mathbb{E}_{i \leftarrow [n]}[x_i] - o(1)\big). \tag{4}$$

To see this, let $\mathbb{E}_{i \leftarrow [n]}[x_i] = \mu$. Under the constraint that $\sum_{i=1}^{n} x_i = \mu \cdot n$ and $x_i \in [0,1]$, the sum $\sum_{i=1}^{n} \frac{1}{n-i+1} \cdot x_i$ takes its smallest possible value when all of the weight is "concentrated" in the first x_i's, i.e. when $x_i = 1$ for all $1 \leq i \leq \mu n$ and $x_i = 0$ for all $\mu n < i \leq n$. In such a case, the sum may be bound as follows

$$\sum_{i=1}^{n} \frac{1}{n-i+1} \cdot x_i = \sum_{i=1}^{\mu n} \frac{1}{n-i+1} = H_n - H_{\lceil (1-\mu)\cdot n \rceil} = \ln(n) - \ln\big((1-\mu)n\big) - o(1) \geq \mu - o(1).$$

This proves (4) and, in particular, the inequality in the lemma for $r = 1$; Let $n \in \mathbb{N}$, $\mathbf{w}, \mathbf{w}' \in \Sigma^n$ and $i^* \in [n]$. Let x_i, for $i \in [i^*]$, be the binary variable that takes 1 if and only if $\mathbf{w}_i \neq \mathbf{w}'_i$. Then, it holds, by (3) and (4), that

$$\Delta_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') \geq \frac{H_{n-i^*}}{H_n} \cdot \Delta_{\mathsf{S}}(\mathbf{w}_{\geq i^*}, \mathbf{w}'_{\geq i^*}) \geq \frac{1}{H_n} \cdot \Delta_{\mathsf{H}}(\mathbf{w}_{\geq i^*}, \mathbf{w}'_{\geq i^*}) - o(1).$$

We finish by showing the lemma holds for any $r > 1$ by induction. Let $N = n_1 \times \cdots \times n_r$, $\mathbf{w}, \mathbf{w}' \in \Sigma^N$ and $\mathbf{i}^* = (i_1^*, \ldots, i_r^*) \in [N]$ be such that $\Delta_{\mathsf{H}}(\mathbf{w}_{\geq \mathbf{i}^*}, \mathbf{w}'_{\geq \mathbf{i}^*}) = \delta$. Then, by definition,

$$\mathbb{E}_{i_1 \leftarrow [n_1 - i_1^* + 1]}[\Delta_{\mathsf{H}}(\mathbf{w}_{\geq \mathbf{i}^*}[pl_{(1,i_1)}], \mathbf{w}'_{\geq \mathbf{i}^*}[pl_{(1,i_1)}])] = \delta,$$

(recall Definition 3.10) and, by the inductive hypothesis, it holds that

$$\mathbb{E}_{i_1 \leftarrow [n_1 - i_1^* + 1]}[\Delta_{\mathsf{S}}(\mathbf{w}_{\geq \mathbf{i}^*}[pl_{(1,i_1)}], \mathbf{w}'_{\geq \mathbf{i}^*}[pl_{(1,i_1)}])] \geq \frac{1}{H_{(n_2 - i_2 + 1) \times \cdots \times (n_r - i_r + 1)}} \cdot \delta - o(1).$$

We apply (4) with the variables $x_{i_1} = \{\Delta_{\mathsf{S}}(\mathbf{w}_{\geq \mathbf{i}^*}[pl_{(1,i_1)}], \mathbf{w}'_{\geq \mathbf{i}^*}[pl_{(1,i_1)}])\}$ to obtain

$$\mathbb{E}_{i_1 \leftarrow \sigma_{n_1 - i^* + 1}}[\Delta_{\mathsf{S}}(\mathbf{w}_{\geq \mathbf{i}^*}[pl_{(1,i_1)}], \mathbf{w}'_{\geq \mathbf{i}^*}[pl_{(1,i_1)}])] \geq \frac{1}{H_{N-\mathbf{i}^*+\mathbf{1}}} \cdot \delta - o(1),$$

and, by the definition of Δ_{S} and (3), finish the proof of the lemma.

□

Next, we utilize the bound from above, in fact its special case with $r = 1$, to prove that high tree distance at any line (recall Definition 4.6) translates to high suffix distance. As a corollary, this shows that our tensor codes from Section 4 guarantee high suffix distance between the codewords.

Lemma 5.3. *Let $r \in \mathbb{N}$. If an r-dimensional code C has Δ_{T}-distance $\delta : \mathbb{N} \to [0,1]$ at any line, then it has Δ_{S}-distance $\delta^r : (n_1, \ldots, n_r) \mapsto H_N^{-1} \cdot \prod_{d=1}^{r} \delta(n_d) - o(1)$.*

Proof. For ease of notation, let us assume δ is constant. The proof is identical in the general case. Let $\mathbf{w}$ and $\mathbf{w}' \in C$ be two distinct codewords of size N. We prove by induction, that tree distance at all lines means that suffix distance at all $(k-1)$-dimensional hyperplanes where disagreement occurs translates into suffix distance over any k-dimensional hyperplanes containing a disagreement.

More formally, for any $k \in [r]$, let PL_k denote the set of all planes of the form $pl_k(i_{k+1}, \ldots, i_r) = \{(x_1, \ldots, x_k, i_{k+1}, \ldots, i_r) \mid x_d \in [n_d]\}$ over which $\mathbf{w}$ and $\mathbf{w}'$ disagree. Then, we argue

$$\forall\ pl' \in PL_{k-1} :\ \Delta_{\mathsf{S}}(\mathbf{w}_{pl'}, \mathbf{w}'_{pl'}) \geq \delta' \implies \forall\ pl \in PL_k :\ \Delta_{\mathsf{S}}(\mathbf{w}_{pl}, \mathbf{w}'_{pl}) \geq \delta' \cdot \delta / H_{n_k} - o(1). \tag{5}$$

This is sufficient to derive the lemma since suffix distance is δ in any $\ell \in PL_1$ by assumption, and $PL_r = [N]$.

Now, let $pl \in PL_k$. By the tree distance δ at any line in C_{pl} (which is inherited by the distance at any line in C by Remark 4.7), any line ℓ that passes through $\mathbf{i}^*$ along the k^{th} dimension k,[13] that $\mathbf{w}_\ell$ and $\mathbf{w}'_\ell$ have at least δ tree distance. Let P be the set of points on ℓ on which $\mathbf{w}_\ell$ and $\mathbf{w}'_\ell$ disagree. By Lemma 5.2, it holds that $\sum_{\mathbf{p} \in P} \sigma_{|\ell|}(p) \geq \delta / H_{n_k} - o(1)$. Additionally, for any $\mathbf{p} \in P$, let $pl'_{\mathbf{p}}$ be the $(k-1)$-dimensional hyperplane that is orthogonal to the k^{th} dimension and intersects with ℓ at the point $\mathbf{p}$.[14] By definition, $pl'_{\mathbf{p}} \in PL_{k-1}$ By the inductive hypothesis, it holds that $\mathbf{w}'$ and $\mathbf{w}'$ have tree distance at least δ' at any such $pl'_{\mathbf{p}}$. Hence, the tree distance between the codewords at pl may be bound as follows

$$\Delta_{\mathsf{S}}(\mathbf{w}_{pl}, \mathbf{w}'_{pl}) = \sum_{\mathbf{p} \in P} \sigma_{|\ell|}(p) \cdot \Delta_{\mathsf{S}}(\mathbf{w}[pl'_{\mathbf{p}}], \mathbf{w}'[pl'_{\mathbf{p}}]) \geq \delta' \cdot \delta / H_{n_k} - o(1)$$

(recall Definition 5.1). □

5.2 The Local Test

Equipped with the new notion of suffix distance, we state our theorem regarding the local testability of tensor tree codes. To prove the theorem, we follow the lines of the proof from Viderman [Vid15], who shows a similar result for the standard notion of local testability of tensor codes. More specifically, he relies on the fact that the tensor product of a code with Hamming distance δ satisfies Hamming distance δ at any line to derive the local testability of tensor codes of order $r \geq 3$. We are interested in showing an analogous statement for suffix distance; Since our tensors satisfy good tree distance at any line, they have good suffix distance at any line as well (by Lemma 5.2). We use the algebraic nature of suffix distance, which is shared by Hamming distance, to apply the same ideas and prove our version of the theorem presented below.

Theorem 5.4 (Local Testability for Suffix Distance)**.** *Let $r \geq 3$. Let C be a tree code with distance $\Delta_{\mathsf{T}}(C) = \delta$ and let C^r be its r-fold tensor from Definition 4.4. Then, C^r is a $(\Delta_{\mathsf{S}}, n^{r-1}, \delta^{r-1}/2r^2 H_N)$-robust LTC via a plane tester, where $n = L_\infty(N)$.*

While the theorem above gives a plane tester with complexity $n^{(r-1)/r}$ (see Definition 3.10), one can bootstrap this test to a test of complexity merely $n^{2/r}$ since it satisfies the powerful notion of robust LTC (see Definition 3.4 and Proposition 3.11).

Corollary 5.5. *Let $r \geq 3$. Let C be a tree code with distance $\Delta_{\mathsf{T}}(C) = \delta$ and let C^r be its r-fold tensor. Then, C^r is a $(\Delta_{\mathsf{S}}, n^2, (\delta^{r-1}/2r^2 H_N)^r)$-robust LTC, where $n = L_\infty(N)$.*

[13] Meaning that all coordinates in ℓ share the same value at all dimensions except the k^{th}

[14] All coordinates in $pl'_{\mathbf{p}}$ share the same value at the d^{th} dimension which is equal to the value in $\mathbf{p}$.

$T_N(\mathbf{w})$:

1. Sample $d \leftarrow [r]$
2. Output $pl \leftarrow PL_d$, sampled with probability $\sigma(pl)$

Figure 5: The tester T

Proof of Theorem 5.4 Let $N = n_1 \times \cdots \times n_r$. We define, for any plane $pl \in PL(N)$ (see Definition 3.10), the sum $\sigma(pl) = \sum_{p \in pl} \sigma(p)$ (see Definition 5.1). Notice that σ is a probability density function over $PL_d = \{pl_{(d,i)} \mid i \in [n_d]\}$ for any $d \in [r]$. In particular, it holds that $\sigma(pl_{(d,i)}) = \sigma_{n_d}(i)$.

We define the *plane tester* $T = \{T_N\}$ in Figure 5 and show that it satisfies the desired robustness.

Fix a word $\mathbf{w} \in \Sigma^N$. For any plane $pl \in PL$, let $\mathbf{w}^*_{pl} \in C^{r-1}$ be the closest codeword to $\mathbf{w}_{pl}$ in Δ_{S}-distance.

We define the *disagreement matrix* $E \in \{0,1\}^N$ as follows. For any point $p \in [N]$,

$$E[p] = 1 \iff \exists pl_1, pl_2 \in PL : \; p \in pl_1 \cap pl_2 \text{ and } \mathbf{w}^*_{pl_1}[p] \neq \mathbf{w}^*_{pl_2}[p].$$

Additionally, we define the *fix matrix* $F \in \{0,1\}^N$ at any point $p \in [N]$ by

$$F[p] = 1 \iff E[p] = 0 \wedge \mathbf{w}[p] \neq a_p,$$

where $a_p = \mathbf{w}^*_{pl}$ for any $pl \in PL$ that contain p (notice that a_p is immaterial to pl when $E[p] = 0$).

Claim 5.6. *It holds, for $pl \leftarrow T$, that*

$$\mathbb{E}_{pl}[\Delta_{\mathsf{S}}(\mathbf{w}_{pl}, C^{r-1})] \geq \frac{1}{r} \cdot \omega_{\mathsf{S}}(E) + \omega_{\mathsf{S}}(F).$$

Proof. By the definition of Δ_{S} (Definition 5.1),

$$\mathbb{E}_{pl}[\Delta_{\mathsf{S}}(\mathbf{w}_{pl}, C^{r-1})] = \Pr_{pl, p \leftarrow \sigma_{|pl|}} \left[\mathbf{w}^*_{pl}[p] \neq \mathbf{w}[p]\right].$$

We can therefore think of the expected distance between $\mathbf{w}_{pl}$ and the corresponding restricted code as the probability of disagreement between $\mathbf{w}^*_{pl}$ and $\mathbf{w}$ at a random point $p \leftarrow \sigma_{|pl|}$, for a random plane pl chosen by the tester. By construction and definition of σ, it is not hard to see that this is equivalent to sampling a point $p \leftarrow \sigma_N$, then sampling a uniformly random plane pl from the set of r planes that contain p (the probability of getting any pl given p can be calculated as $\Pr[pl \mid p] = \Pr[pl] \cdot \Pr[p \mid pl] / \Pr[p] = 1/r$). Hence, for $pl \leftarrow T$,

$$\mathbb{E}_{pl}[\Delta_{\mathsf{S}}(\mathbf{w}_{pl}, C^{r-1})] = \Pr_{p \leftarrow \sigma_N, d \leftarrow [r]} \left[\mathbf{w}^*_{pl'}[p] \neq \mathbf{w}[p]\right], \qquad \text{where } pl' = pl_{(d,p_d)}.$$

Let us split the probability space by the following two disjoint cases: (i) We sample p such that $E[p] = 1$, in which case the event holds whenever we choose the plane that does not agree with $\mathbf{w}$ at p (there must exist such a plane since there are at least two planes that do not agree at p). Recall such a plane is sampled uniformly at random and, therefore, the probability to "catch the

error" in this case is at least $1/r$. (ii) We sample p such that $E[p = 0]$. For such a point p, all planes pl agree on the same value for $\mathbf{w}^*_{pl}[p]$ and, therefore, the event occurs if and only if this value is different from $\mathbf{w}[p]$, namely if $F[p] = 1$. The proof of the claim is then complete by law of total probability. □

Our strategy at this point is two-fold: First, we argue that, by the high tree distance of C, the weight in E must be concentrated in few "heavy" planes and that, by removing these planes, one ends up with an all-zero E. Second, we leverage this concentration property of E to derive a correlation between the weight of disagreements $\omega_{\mathsf{S}}(E)$ and the weight $\omega_{\mathsf{S}}(F)$ (which bounds robustness due to Claim 5.6), and the distance of $\mathbf{w}$ from the code C^r. Intuitively speaking, when all disagreements are concentrated in few planes, then there is a proportionally big subspace inside $\mathbf{w}$ where all planes agree on a "global" closest codeword in the corresponding subcode. Then, all is required to make $\mathbf{w}$ a codeword, is to fix this subspace (this corresponds to points in F) then change the values in the disagreements (i.e. E) accordingly.

Let us begin with the following bound on $\omega_{\mathsf{S}}(F)$.

Claim 5.7. *Let $S_1, \ldots, S_r$ be subsets of planes where, for all $d \in [r]$, $S_d \subseteq PL_d$ and $E_{S_1 \times \cdots \times S_r} = \mathbf{0}$. Then, it holds that*

$$\omega_{\mathsf{S}}(F) \geq \Delta_{\mathsf{S}}(\mathbf{w}, C^r) - \sum_{d=1}^{r} \sigma_N(\overline{S_d})$$

where, for any $d \in [r]$, $\sigma_N(\overline{S_d}) = \sum_{pl \in PL_d \setminus S_d} \sigma_N(pl)$.

Proof. Denote $S = S_1 \times \cdots \times S_r$. For any $p \in S$, let a_p be the value such that $\mathbf{w}^*_{pl}[p] = a_p$ for all plane pl that contains p (recall all planes agree at p). Observe that $A_S = \{a_p\}_{p \in S}$ is a codeword in $C_{S_1} \otimes \cdots \otimes C_{S_r}$. Therefore, one can "fix" $\mathbf{w}$ to be a codeword by changing the values in any point $p \in F$ to take $\mathbf{w}_S$ to A_S and, then, changing the values in any point $p \notin S$ accordingly to match a codeword that extends A_S (there always exist such since $C_{S_1} \otimes \cdots \otimes C_{S_r} = C^r_S$). Therefore,

$$\Delta_{\mathsf{S}}(\mathbf{w}, C^r) \leq \omega_{\mathsf{S}}(F) + \sum_{p \notin S} \sigma_N(p) \leq \omega_{\mathsf{S}}(F) + \sum_{d=1}^{r} \sum_{p \notin S_d} \sigma_N(p) = \omega_{\mathsf{S}}(F) + \sum_{d=1}^{r} \sum_{pl \in PL_d \setminus S_d} \sigma_N(pl).$$

□

It remains to show that E is indeed concentrated. Let us first formalize the notion of a heavy plane.

Definition 5.8. *We say that a plane $pl \in PL_d$ is* heavy *if $\omega_{\mathsf{S}}(E_{pl}) \geq (H_{n_d}/H_N) \cdot \delta^{r-1}/2$.*

In the next key lemma, we prove that the heavy planes contain all points of disagreement.

Lemma 5.9. *For any point $p \in [N]$, if $E[p] = 1$ then p is on a heavy a plane.*

Proof. Let $pl_1 = pl_{(1,i_1)}$ and $pl_2 = pl_{(2,i_2)}$ be, without loss of generality, the two planes that do not agree at p (see Definition 5.8). Let $pl' = pl_1 \cap pl_2$ be the $(r-2)$-dimensional plane in their intersection. Recall C^r and, in particular, C^{r-2}, satisfy tree distance δ at any line (see Corollary 4.8). Since $\mathbf{w}^*_{pl_1} \neq \mathbf{w}^*_{pl_2}$, and by Lemma 5.3, this implies suffix distance $\eta \cdot \delta^{r-2} - o(1)$ between $\mathbf{w}^*_{pl_1}$ and $\mathbf{w}^*_{pl_2}$ at pl', where $\eta = H_{n_1} H_{n_2}/H_N$. That is, letting

$$B = \{p' \in pl' \mid \mathbf{w}^*_{pl_1}[p'] \neq \mathbf{w}^*_{pl_2}[p']\},$$

it holds that $\sigma_{|pl'|}(B) = \sum_{p' \in B} \sigma_{|pl'|}(p'_{-1,2}) \geq \eta \cdot \delta^{r-2}$, where $p'_{-1,2}$ is the $(r-2)$-dimensional coordinate that is the projection of p' on pl' (i.e. obtained by removing the first two dimensions).

For any $p' \in B$, let $pl_3(p') = pl_{3,i_3}$ be the unique such plane that contains p', i.e. $p' \in pl_3(p')$. For $b \in \{1,2\}$, let

$$B_b = \{p' \in B \mid \mathbf{w}^*_{pl_b}[p'] \neq \mathbf{w}^*_{pl_3(p')}[p']\},$$

and assume, w.l.o.g., that $\sigma_{|pl'|}(B_1) \geq \sigma_{|pl'|}(B)/2 \geq \eta \cdot \delta^{r-2}/2 - o(1)$ (it holds that $\sigma_{|pl'|}(B) \leq \sigma_{|pl'|}(B_1) + \sigma_{|pl'|}(B_2)$).

Denote by $\ell(p')$, for any $p' \in B_1$, the one-dimensional line in the intersection $pl_1 \cap pl'_3(p')$ that is orthogonal to pl_2. Namely, letting $p' = (i_1, i_2, p'_3, \ldots, p'_r)$, we define $\ell(p') = \{(i_1, i, p'_3, \ldots, p'_r) \mid i \in [n_2]\}$. Again, by the tree distance of C at any line, it holds that $\Delta_{\mathsf{S}}(\mathbf{w}^*_{pl_1}[\ell(p')], \mathbf{w}^*_{pl_3(p')}[\ell(p')]) \geq \eta_2 \cdot \delta - o(1)$ for any $p' \in B_1$, where $\eta_2 = 1/H_{n_2}$. Hence, letting

$$B_{p'} = \left\{p'' \in \ell(p') \mid \mathbf{w}^*_{pl_1}[p''] \neq \mathbf{w}^*_{pl_3(p')}[p'']\right\},$$

it holds that $\sigma_{|\ell(p')|}(B_{p'}) = \sum_{p'' \in B_{p'}} \sigma_{|\ell(p')|}(p''_2) \geq \eta_2 \cdot \delta$, where p''_2 is the projection of p'' to the second dimension, i.e. the position of p'' in $\ell(p')$.

Now, since any point in any $B_{p'}$ is a point of disagreement, that is a point where E has 1, we obtain that

$$\omega_{\mathsf{S}}(E_{pl_1}) = \sum_{p' : E[p']=1} \sigma_{|pl_1|}(p') \geq \sum_{p' \in B_1} \sum_{p'' \in B_{p'}} \sigma_{|pl_1|}(p'') = \sum_{p' \in B_1} \sum_{p'' \in B_{p'}} \sigma_{|pl'|}(p''_{-1,2}) \cdot \sigma_{|\ell(p')|}(p''_2)$$

Next, observe that any p' and $p'' \in B_{p'}$ are equal at all dimensions but the second (by definition of $\ell(p')$). Therefore,

$$\omega_{\mathsf{S}}(E_{pl_1}) \geq \sum_{p' \in B_1} \sigma_{|pl'|}(p'_{-1,2}) \cdot \sum_{p'' \in B_{p'}} \sigma_{|\ell(p')|}(p''_2) \geq \sum_{p' \in B_1} \sigma_{|pl'|}(p'_{-1,2}) \cdot \eta_2 \delta \geq \eta_1 \eta_2 \cdot \frac{\delta^{r-1}}{2} = \frac{H_{n_1}}{H_N} \cdot \frac{\delta^{r-1}}{2},$$

and pl_1 is heavy.

□

Lemma 5.9 immediately gives the following corollary.

Corollary 5.10. *There exist subsets of planes $S_1, \ldots, S_r$ where $S_d \subseteq PL_d$ for all $d \in [r]$, such that $E_{S_1 \times \cdots \times S_r} = \mathbf{0}$ and*

$$\sum_{d=1}^{r} \sigma_N(\overline{S_b}) \leq \frac{2r \cdot H_N}{\delta^{r-1}} \cdot \omega_{\mathsf{S}}(E)$$

.

Proof. For any $d \in [r]$, let $S_d = \{pl \in PL_d \mid pl \text{ is not a heavy plane}\}$. By Lemma 5.9, it holds that $E_{S_1 \times \cdots \times S_r} = \mathbf{0}$. Additionally, since any point $p \in [N]$ resides on at most r planes in PL (one in each direction), then

$$\omega_{\mathsf{S}}(E) \geq \frac{1}{r} \cdot \sum_{d=1}^{r} \sum_{pl \in PL_d \setminus S_d} \sigma_N(pl) \cdot \omega_{\mathsf{S}}(pl) \geq \frac{\delta^{r-1}}{2r \cdot H_N} \cdot \sum_{d=1}^{r} \sigma_N(\overline{S_b}).$$

□

Now, by plugging Corollary 5.10 in Claim 5.7, we obtain

$$\Delta_{\mathsf{S}}(\mathbf{w}, C^r) \leq \frac{2r \cdot H_N}{\delta^{r-1}} \cdot \omega_{\mathsf{S}}(E) + \omega_{\mathsf{S}}(F) \leq \frac{2r^2 \cdot H_N}{\delta^{r-1}} \cdot \left(\frac{1}{r} \cdot \omega_{\mathsf{S}}(E) + \omega_{\mathsf{S}}(F) \right).$$

By further plugging in Claim 5.6, we complete the proof of Theorem 5.4.

6 Flattening Tensors to One Dimension

In the previous section, we have presented a construction of tensor tree codes. These codes are defined through an encoding function that is online in the high-dimensional sense (see Definition 4.1). Additionally, we have demonstrated they satisfy meaningful notions of distance: tree distance at all lines, which is an adaptation of tree distance to the high-dimensionality of time underlying the online encoding of tensors, and, as a consequence, the weaker notion of suffix distance. Further, by generalizing the analysis from [Vid15], we have shown that these codes are locally testable w.r.t. suffix distance and, therefore, w.r.t. to tree distance at all lines.

Recall, however, that our ultimate goal is to construct a tree code that is locally testable. While tensor codes alone give a solution to the problem of local testability in the standard setting, what we have achieved so far using tensors does not entirely fulfill our ambitions for tree codes. Despite being a promising first step, it is not clear how to turn online encoding over the high-dimensional space to online encoding in the standard tree-code sense and, at the same time, to convert the high-dimensional distance and local testability properties to their standard counterparts (namely tree distance and LTC with respect to it).

With this goal in mind, we propose a method to transform the tensor tree code (in fact, any high-dimensional tree code) to a standard tree code. The main component in our scheme is a *traversal procedure* over the high-dimensional space of coordinates $\mathbb{N}^r$. Through the traversal, we define an order over the coordinates in $\mathbb{N}^r$ and, consequently, derive an embedding of their space to the one-dimensional space $\mathbb{N}$ (where in the i^{th} location we embed the i^{th} coordinate reached by the traversal). We then use the embedding in hand to produce an online encoding function which, at time $i \in \mathbb{N}$, produces the encoding, under the high-dimensional code, that corresponds to the high-dimensional coordinate embedded at i. We refer to the resulted code by the *flattened code*.

In the latter part of this section, we show that not only our traversal enables us to flatten tensor tree codes and achieve the desired notion of online encoding, but that, additionally, the flattened code also turns out to be locally testable w.r.t. standard tree distance! Specifically, via thorough analysis, we show that the code can be locally tested by applying the local test for the underlying the high-dimensional code (from Section 4) over random parts of the flattened codeword.

6.1 Traversing the High-Dimensional Space

We begin with describing the traversal over the high-dimensional space $\mathbb{N}^r$ that stands at the heart of our flattening transformation.

For the traversal to fit into the outline described above, it has to satisfy a crucial property; Namely, that it reaches coordinate $\mathbf{i} \in \mathbb{N}^r$ only after it has reached any coordinate $\mathbf{i}' \leq \mathbf{i}$ (recall the partial order presumed over $\mathbb{N}^r$). Notice that this is important to ensure that when the (flattened) encoding function will arrive at the $\mathbf{i}^{th}$ coordinate it is able to call the underlying high-dimensional encoder, which presumes all values in the sub-cube $[\mathbf{i}] = \{\mathbf{i}' \mid \mathbf{i}' \leq \mathbf{i}\}$ are available for encoding the

$\mathbf{i}^{th}$ codeword symbol. Briefly put, we require the traversal to be *monotone* w.r.t. the partial order over $\mathbb{N}^r$.

Before we present our traversal, let us suggest a new useful way for representing coordinates in high dimensions. We will use this representation in the construction and in many parts of its analysis. As we will see throughout the section, it facilitates the understanding of how our traversal behaves since it is strongly connected to the order it defines.

Definition 6.1 (Histogram Representation of Coordinates)**.** *Let $r \in \mathbb{N}$ and let $\mathbf{i} = (i_1, \ldots, i_r) \in \mathbb{N}^r$ be a coordinate. Let $t_1 > t_2 > \cdots > t_\ell$ be the list of distinct integers in $(i_1, \ldots, i_r)$ in descending order. For any $k \in [\ell]$, let $J_k = \{j \in [r] \mid i_j = t_k\}$. We refer to $\big((t_1, J_1), \ldots, (t_\ell, J_\ell)\big)$ as the* histogram representation *of $\mathbf{i}$. For simplicity, since J_ℓ is determined by $J_1, \ldots, J_{\ell-1}$, we sometimes write $\big((t_1, J_1), \ldots, (t_{\ell-1}, J_{\ell-1}), (t_\ell, *)\big)$.*

The Traversal. We define our traversal over the r-dimensional coordinate space via the recursive procedure in Fig. 6. We refer the reader to the overview in Section 2 for an intuitive description of the traversal.

The Traversal $\Pi^r(t)$, from 1^r to t^r

For $d = 0, \ldots, r$,

For all $J \in \binom{r}{d}$ in lexicographic order[a],

1. Let $\mathbf{t}_J^{(0)} = ((t, J), (1, *))$.
2. Let $\mathbf{t}_J^{(1)} = ((t, J), (t-1, *))$.
3. Walk recursively via $\Pi^{r-d}(t-1)$, on the $(r-d)$-dimensional subspace consisting of all dimensions $[r] \setminus J$, from $\mathbf{i}_J^{(0)}$ to $\mathbf{i}_J^{(1)}$.

[a] We require that the order satisfies Postulate 6.2 but is otherwise arbitrary.

Figure 6: Traversal over $\mathbb{N}^r$.

Observe that the first recursive call in $\Pi^r(t)$ is made to $\Pi^r(t-1)$. Consequently, the procedure above defines an infinite traversal over $\mathbb{N}^r$ where, for any $t \in \mathbb{N}$, the first t^r steps constitute the traversal $\Pi^r(t)$. We denote the infinite traversal simply by Π^r. Regardless, any coordinate $\mathbf{i} \in \mathbb{N}^r$ is reached by the finite traversal $\Pi^r(L_\infty(\mathbf{i}))$.

We define the function $\pi^r : \mathbb{N} \to \mathbb{N}^r$ that maps any n to the n^{th} coordinate in the order induced by the walk Π^r (i.e. $\pi^r(n)$ is the n^{th} coordinate that the walk reaches). We omit r when it is unambiguous by context. Further, we denote by $\pi^{-1} : \mathbb{N}^* \to \mathbb{N}$ the inverse function, that takes any coordinate $N \in \mathbb{N}^r$ and maps it to its order in the walk Π^r.

Lastly, the traversal Π^r defines an order over $\mathbb{P}([r])$: For any two subsets $J, J' \subseteq [r]$, we denote $J \prec J'$ if and only if the loop in Π^r (see Fig. 6) iterates over J before it does over J', namely if and only if $|J| < |J'|$ or $|J| = |J'|$ and J precedes J' in the arbitrary lexicographic order chosen

in the walk. We assume the following regarding the lexicographic order over $\mathbb{P}([r])$ defined by the traversal.

Postulate 6.2. *For any J, J' and $j' \in J'$, if $J \succ J'$, then there exists $j \in J$ such that either $J \setminus \{j\} \succ J' \setminus \{j'\}$, or $J \setminus \{j\} = J' \setminus \{j'\}$ and $\{j\} \succ \{j'\}$.*

Observe that many natural lexicographic orders over $\mathbb{P}([r])$ (e.g. standard lexicographic order over strings representing the sorted sets) satisfy Postulate 6.2.

Given the above lexicographic order over subsets of dimensions, we make the following observation, which connects the order obtained by the traversal Π^r to a lexicographic order over the histogram representation of the coordinates of $\mathbb{N}^r$.

Proposition 6.3. *Let $\mathbf{i}, \mathbf{i}' \in \mathbb{N}^r$ be two coordinates and let $((t_1, J_1), \ldots, (t_\ell, J_\ell)), ((t'_1, J'_1), \ldots, (t'_{\ell'}, J'_{\ell'}))$ be their respective histogram representations (see Definition 6.1). It holds that $\pi^{-1}(\mathbf{i}) < \pi^{-1}(\mathbf{i}')$ if and only if $((t_1, J_1), \ldots, (t_\ell, J_\ell))$ is smaller than $((t'_1, J'_1), \ldots, (t'_\ell, J'_\ell))$ in an element-by-element comparison, where the "most significant bit" corresponds to (t_1, J_1), and (t, J) is smaller than (t', J') if and only if $t < t'$, or $t = t'$ and $J \prec J'$.*

With the above proposition, it immediately follows that our traversal is monotone.

Proposition 6.4. *For any $r \in \mathbb{N}$, the function π^{-1} is* monotone*. Namely, for any $\mathbf{i}, \mathbf{i}' \in \mathbb{N}^r$, it holds that if $\mathbf{i} \leq \mathbf{i}'$ then $\pi^{-1}(\mathbf{i}) \leq \pi^{-1}(\mathbf{i}')$.*

Equipped with the traversal, we are prepared to define the flattened code. For any string $x \in \Sigma^n$, we denote by $\mathbf{lift}^r(x)$ the *lifting of x to r dimensions*, namely the (possibly partial) r-dimensional tensor where, for any $\mathbf{i} \in \mathbb{N}^r$ such that $\pi^{-1}(\mathbf{i}) \leq n$, the $\mathbf{i}^{th}$ symbol is equal to $x_{\pi^{-1}(\mathbf{i})}$. Note that $\mathbf{lift}^r(x)$ is partial unless $n = t^r$ for some $t \in \mathbb{N}$. However, by Proposition 6.4, $\mathbf{lift}^r(x)$ is defined at all $\mathbf{i} \leq \pi^r(|x|)$. Our construction is as follows.

Construction 6.5. *Let $C = \{C_N : \Sigma_{\mathsf{in}}^N \to \Sigma_{\mathsf{out}}\}_{N \in \mathbb{N}^r}$ be an r-dimensional tree code. The* flattening *of C, denoted by $\overline{C} = \{\overline{C}_n\}_{n \in \mathbb{N}}$, is the function ensemble where, for any $n \in \mathbb{N}$, $\overline{C}_n : \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}$ is defined as follows*

$$\overline{C}_n(m) = C_{\mathbf{n}}(\mathbf{m}_{\leq \mathbf{n}}),$$

where $\mathbf{n} = \pi^r(n)$ and $\mathbf{m} = \mathbf{lift}^r(m)$.

6.2 Local Testability of the Flattened Code

In this section we prove a local testability theorem for our flattened code from Construction 6.5. We show that, if the base code C is locally testable w.r.t. suffix distance, then the flattened code satisfies our sought local testability notion, that is, local testability w.r.t. tree distance.

Theorem 6.6 (From Δ_{S}-LTC to Δ_{T}-LTC)**.** *Let C be an r-dimensional tree code that has Δ_{S}-distance δ_C and is $(\Delta_{\mathsf{S}}, q, \epsilon)$-strong LTC. Then, $\overline{C}$ is $(\Delta_{\mathsf{T}}, (\epsilon \cdot \delta_C)^{-1} \cdot \log^3 n \cdot q(N), 1 - H_r/r)$-LTC, where $N = (n^{1/r} + 1, \ldots, n^{1/r} + 1)$.*

The Tester. Let $T = \{T_N\}_{N \in \mathbb{N}^r}$ be the q-query tester for C that gives the strong LTC property of the code (see Definition 3.4). Our goal is to construct a tester $\overline{T}$ which, given a word $w \in \Sigma^n$, is able to detect whether its lifting to r dimensions, i.e. $\mathbf{w} = \mathbf{lift}^r(w) \in \Sigma^I$, is in the code C_I (here, $I = \{\pi^r(i) \mid i \in [n]\}$). While we are equipped with a tester for C, it expects as input a rectangular (i.e. non-partial) tensor and, therefore, we cannot apply it directly over $\mathbf{w}$, which is partial unless n is an integral r^{th} power.

Our idea is to apply the tester T over a random maximal sub-cube of $\mathbf{w}$; We consider the set of *endpoints* of $\mathbf{w}$, namely all maximal coordinates in I, and let the tester $\overline{T}$ apply T over the sub-cube $\mathbf{w}_{[\mathbf{e}]}$ for a uniformly random endpoint $\mathbf{e}$. To prove that this gives local testability, we show that if w is far from the code $\overline{C}$ in tree distance, then there exists at least one endpoint $\mathbf{e}$ such that $\mathbf{w}_{[\mathbf{e}]}$ is far from C in *suffix distance*. This is sufficient given we additionally bound the number of endpoints by $r \cdot 2^r$.

We begin by defining the set of endpoints corresponding to the coordinates of a partial tensor.

Definition 6.7 (Endpoints of a Partial Tensor)**.** *Let $r \in \mathbb{N}$ and let $I \subseteq \mathbb{N}^r$ be a monotone set of coordinates (see Definition 3.1). We define the set of* endpoints *of I as follows*

$$E(I) = \{\mathbf{e} \in \mathbb{N}^r \mid \mathbf{e} \in I,\ \forall\ \mathbf{i} \geq \mathbf{e}, \mathbf{i} \neq \mathbf{e} :\ \mathbf{i} \notin I\}.$$

We now define the tester $\overline{T} = \{\overline{T}_n\}_{n \in \mathbb{N}}$ for $\overline{C}$ in Fig. 7.

$\overline{T}_n(w \in \Sigma^n)$:

1. Let $\mathbf{w} = \mathbf{lift}^r(w)$ and let $I = \{\pi^r(i) \mid i \in [n]\}$
2. Sample endpoint $\mathbf{e} \leftarrow E(I)$ uniformly at random and output $T_{\mathbf{e}}(\mathbf{w}_{[\mathbf{e}]})$.

Figure 7: The tester $\overline{T}_n$

Proof of Theorem 6.6. As mentioned above, to argue soundness of the test $\overline{T}$, we show that if w is far from $\overline{C}$ in tree distance, then there must exist an endpoint that gives high suffix distance over the induced sub-cube. For such an argument not to incur an undesirable loss in soundness, we must additionally bound the number of endpoints.

As a first step, let us take a closer look at the set of endpoints and its structure. Let $n \in \mathbb{N}$ and $\mathbf{n} = \pi^r(n)$, and let $((t_1, J_1), \ldots, (t_{r'}, J_{r'}))$ be the histogram representation of $\mathbf{n}$ (see Definition 6.1). Recall that the point $\mathbf{n}$ is reached throughout the traversal $\Pi^r(t_1)$ (see Fig. 6). The histogram representation tells us that, at the point the traversal procedure reaches $\mathbf{n}$, it is inside the recursive call $\Pi^{r-|J_1|}(t_1 - 1)$ corresponding to J_1 and that it has had already returned from all recursive calls corresponding to subsets $J \subseteq [r]$ such that $J \prec J_1$ (recall the lexicographic order induced by Π^r over such subsets). The last coordinate to be reached at a completed recursive branch corresponding to such a J is

$$\mathbf{e}_{1,J} = ((t_1, J), (t_1 - 1, *)).$$

The next recursive call within the "current" recursive branch, corresponding to J_1, that is completed before reaching $\mathbf{n}$ corresponds to input $t = t_2$. In fact, the traversal must have had

completed all recursive calls with input t_2 that correspond to subsets $J \prec J_2$. Similarly to above, the last coordinate to be reached upon the completion of such a "depth-2" recursive call, corresponding to a subset J, is

$$\mathbf{e}_{2,J} = ((t_1, J_1), (t_2, J), (t_2 - 1, *)).$$

In general, the last coordinate reached by a recursive call of depth ℓ, that is completed by the time the traversal reaches $\mathbf{n}$, is of the form

$$\mathbf{e}_{\ell,J} = ((t_1, J_1), \ldots, (t_{\ell-1}, J_{\ell-1}), (t_\ell, J), (t_\ell - 1, *)), \tag{6}$$

for some $J \prec J_\ell$ (which is possibly the empty set). In the following claim, we formally argue that these coordinates, corresponding to the maximal subsets $J \prec J_\ell$ at any level of (non-trivial)[15] recursion, constitute the set of endpoints $E(I)$.

Claim 6.8. *Let $n \in \mathbb{N}$, $\mathbf{n} = \pi^r(n)$, $I = \{\pi^r(i) \mid i \in [n]\}$, and let $((t_1, J_1), \ldots, (t_{r'}, J_{r'}))$ be the histogram of $\mathbf{n}$ (see Definition 6.1). For any $1 \leq \ell < r'$, letting $J_{\geq \ell} = \bigcup_{i=\ell}^{r'} J_i$, let us define the set of* depth-ℓ endpoints *as follows*

$$E_\ell = \{\mathbf{e}_{\ell,J} \mid J \subseteq J_{\geq \ell}, J \prec J_\ell, \text{ and } \forall J' \subseteq J_{\geq \ell} \text{ s.t } J' \prec J_\ell, J' \neq J : J \not\subset J'\},$$

where $\mathbf{e}_{\ell,J}$ is as defined in Eq. (6), and

$$E_{r'} = \{\mathbf{n}\}.$$

Then, it holds that

$$E(I) \subseteq \bigcup_{\ell=1}^{r'} E_\ell \subseteq I(n).$$

Proof. The fact that for any $\mathbf{e}_{\ell,J} \in \bigcup E_\ell$ it holds that $\pi^{-1}(\mathbf{e}) \leq n$ follows immediately from Proposition 6.3. To prove that all endpoints are contained in $\bigcup E_\ell$, we show that for any $\mathbf{i}$ such that $\pi^{-1}(\mathbf{i}) \leq n$, there exists $\mathbf{e} \in \bigcup E_\ell$ such that $\mathbf{i} \leq \mathbf{e}$.

Let $((t'_1, J'_1), \ldots, (t'_{r''}, J'_{r''}))$ be the histogram representation of $\mathbf{i}$. Let ℓ be the smallest integer such that (t'_ℓ, J'_ℓ) is smaller than (t_ℓ, J_ℓ) by the lexicographic order defined in Proposition 6.3 (if no such ℓ exists then $\mathbf{i} = \mathbf{n}$ and we finish). If $t'_\ell < t_\ell$ then $\mathbf{i} \leq \mathbf{e}$ for any $\mathbf{e} \in E_\ell$ (the $(\ell+1)^{th}$ biggest value in $\mathbf{i}$ is at most $t_\ell - 1$). If $t_\ell = t'_\ell$ then it must be the case that $J'_\ell \prec J_\ell$. Letting J'' be the maximal subset such that $J'' \prec J_\ell$ and $J'_\ell \subseteq J''$, it holds that $\mathbf{e}_{\ell,J''} \in E_\ell$ and $\mathbf{i} \leq \mathbf{e}_{\ell,J''}$, hence we finish. □

As a corollary to Claim 6.8, we have shown a bound on the number of endpoints:

$$|E(I)| \leq r \cdot 2^r. \tag{7}$$

In a second preliminary step that is crucial to our proof, we establish a connection between the tree distance between strings and the suffix distance between their lifting to r dimensions. Since we will be working with tensors that are partial, we start by generalizing these metrics to the setting of partial tensors (to generalize the notion of tree distance to "flattened" partial tensors it suffices to say how to flatten such tensors).

[15] Where a trivial recursive call corresponds to $J = \emptyset$.

Definition 6.9 (Flattening of (Partial) Tensors)**.** *Let $r \in \mathbb{N}$ and let $I \subset \mathbb{N}^r$ be a finite subset of coordinates. We denote by Π^r_I the restriction of the traversal Π^r from Fig. 6 to coordinates in I (i.e. Π^r_I traverses I in the order induced by Π^r). We subsequently define the function $\pi^r_I : [|I|] \to I$ that maps any integer $1 \leq i \leq |I|$ to the i^{th} coordinate in the walk Π^r_I (namely, the i^{th} coordinate that is reached by Π^r among the coordinates in I) and its inverse $\pi_I^{-1} : I \to [|I|]$.*

Let $\mathbf{w} \in \Sigma^I$ be a (possibly partial) r-dimensional tensor. We denote by $w = \overline{\mathbf{w}}$ the flattening of $\mathbf{w}$ *to one dimension and define it as the string of length $|I|$ where $w[i] = \mathbf{w}[\pi^r_I(i)]$ for all $1 \leq i \leq |I|$.*

Lastly, let $C \subset \Sigma^*$ be an r-dimensional code and let C_I be its restriction to a subset $I \subset \mathbb{N}^r$. We denote $\overline{C}_I = \{\overline{\mathbf{w}} \mid \mathbf{w} \in C_I\}$.[16]

Definition 6.10 (Average Suffix Distance for Partial Tensors)**.** *Let $r \in \mathbb{N}$ and let $\mathbf{w}, \mathbf{w}' \in \Sigma^I$ be two (possibly partial) r-dimensional tensors. We define the* average suffix distance *between $\mathbf{w}$ and $\mathbf{w}'$ as*

$$\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') = \mathbb{E}_{\mathbf{e} \leftarrow E(I)} \left[\Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, \mathbf{w}'_{[\mathbf{e}]}) \right].$$

where $E(I)$ is the set of endpoints defined in Definition 6.7.

Claim 6.11. *Let $I_1, \ldots, I_m \subset \mathbb{N}^r$ be monotone and let $I = \bigcup_{j=1}^m I_j$. For any $\mathbf{w}, \mathbf{w}' \in \Sigma^I$, letting $\mathbf{w}_j = \mathbf{w}[I_j]$ and $\mathbf{w}'_j = \mathbf{w}'[I_j]$, there exists $j \in [m]$ such that*

$$\Delta_{\mathsf{T}}(w_j, w'_j) \geq \frac{1}{m} \cdot \Delta_{\mathsf{T}}(w, w'),$$

where w, w', w_j, w'_j are the flattening of the respective tensors (see Definition 6.9).

Proof. Let i^* be the smallest integer such that $w_{i^*} \neq w'_{i^*}$. Let $S = \{i \in \mathbb{N} \mid i \geq i^*, \pi^r(i) \in I\}$ and $D = \{i \in S \mid w_i \neq w'_i\}$ and, for any $j \in [m]$, let $S_j = S \cap I_j$ and $D_j = D \cap I_j$. Denote $\delta = \Delta_{\mathsf{T}}(w, w')$. By definition, it holds that $\delta = |D|/|S|$ and $\Delta_{\mathsf{T}}(w_j, w'_j) \geq |D_j|/|S_j|$ (equality occurs only when the smallest element in S_j is also in D_j). Now, by averaging, since $\sum_j |D_j| \geq |D|$, there must exist $j \in [m]$ such that $|D_j| \geq |D|/m$. For that j, it holds that $|D_j|/|S_j| \geq |D|/m|S_j| \geq |D|/m|S| = \delta/m$. □

Proposition 6.12. *Let C be an r-dimensional code with Δ_{S}-distance δ (see Definition 3.3). Then, for any monotone $I \subset \mathbb{N}^r$ (see Definition 3.1) and any distinct $\mathbf{w}, \mathbf{w}' \in C_I$, it holds that $\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') \geq \min_{\mathbf{e} \in E(I)} \delta(\mathbf{e})/|E(I)|$.*

Proof. Let $\mathbf{w}, \mathbf{w}' \in C_I$ be two distinct codewords, and let $\mathbf{e} \in E(I)$ be such that $\mathbf{w}_{[\mathbf{e}]} \neq \mathbf{w}'_{[\mathbf{e}]}$. Then, since $\mathbf{w}_{[\mathbf{e}]}, \mathbf{w}'_{[\mathbf{e}]} \in C$ it holds that $\Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, \mathbf{w}'_{[\mathbf{e}]}) \geq \delta(\mathbf{e})$ and the proposition follows by definition of $\Delta^{\mathsf{avg}}_{\mathsf{S}}$. □

Lemma 6.13. *Let $I \subset \mathbb{N}^r$ be monotone (see Definition 3.1) and let $m = \max_{\mathbf{i} \in I} L_\infty(\mathbf{i})$. Then, for any $\mathbf{w}, \mathbf{w}' \in \Sigma^I$, it holds that*

$$\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}, \mathbf{w}') \geq \frac{1}{r \cdot (2H_m)^r \cdot |E(I)|^2} \cdot \Delta_{\mathsf{T}}(w, w') - o(1),$$

where $w = \overline{\mathbf{w}}$ and $w' = \overline{\mathbf{w}}'$.

[16] Note this is consistent with the notation used in Construction 6.5 and therefore introduces no ambiguity.

Proof. Let $i^* \in [|I|]$ be the smallest integer such that $w_{i^*} \neq w'_{i^*}$. Let $\mathbf{i}^* = \pi_I^r(i^*)$ and let $S = \{\mathbf{i} \in I \mid \pi_I^{-1}(\mathbf{i}) \geq i^*\}$. Let $S^* = \{\mathbf{s} \in S \mid \forall \mathbf{i} \in S, \mathbf{i} \not\leq \mathbf{s}\}$. In other words, S^* are the start points of all maximal r-dimensional suffixes that are contained in the one-dimensional suffix starting at the first disagreement between w and w'.

We argue that $|S^*| \leq r \cdot 2^r$. To see this, let $((t_1^*, J_1^*), \ldots, (t_{\ell^*}^*, J_{\ell^*}^*))$ be the histogram representation of $\mathbf{i}^*$ (see Definition 6.1), and let us define the following two sets of coordinates (via their histogram representations):

$$S^*_{\prec} = \bigcup_{k=1}^{\ell^*} \left\{ \left((t_1^*, J_1^*), \ldots, (t_{k-1}^*, J_{k-1}^*), (t_k^* + 1, J'_k), (1, *)\right) \mid J'_k \prec J_k^* \right\}$$

and

$$S^*_{\succ} = \bigcup_{k=1}^{\ell^*} \left\{ \left((t_1^*, J_1^*), \ldots, (t_{k-1}^*, J_{k-1}^*), (t_k^*, J'_k), (1, *)\right) \mid J'_k \succ J_k^* \right\}.$$

It is evident that $|S^*_{\prec} \cup S^*_{\succ} \cup \{\mathbf{i}^*\}| \leq r \cdot 2^r$. We now show that this union must contain all coordinates in S^*. To that end, it suffices to show that any coordinate in S is contained in a suffix that starts at some $\mathbf{s} \in S^*_{\prec} \cup S^*_{\succ}$.

Let $\mathbf{i} \in S \backslash \{\mathbf{i}^*\}$ be such a coordinate and let $((t_1, J_1), \ldots, (t_\ell, J_\ell))$ be its histogram representation. Since $\pi_I^{-1}(\mathbf{i}) > \pi_I^{-1}(\mathbf{i}^*)$ and, therefore, $\pi^{-1}(\mathbf{i}) > \pi^{-1}(\mathbf{i}^*)$, then it holds by Proposition 6.3 that the histogram of $\mathbf{i}$ is bigger than $((t_1^*, J_1^*), \ldots, (t_{\ell^*}^*, J_{\ell^*}^*))$ in lexicographic order (that is defined in Proposition 6.3). Let k be the first location where $(t_k, J_k) \neq (t_k^*, J_k^*)$. There are two possible cases:

- $t_k > t_k^*$ and $J_k \prec J_k^*$: In which case the coordinate $\mathbf{s} \in S^*_{\prec}$ represented by the histogram $((t_1^*, J_1^*), \ldots, (t_k^* + 1, J_k), (1, *))$ satisfies $\mathbf{s} \leq \mathbf{i}$.
- $t_k \geq t_k^*$ and $J_k \succ J_k^*$: Then, the coordinate $\mathbf{s} \in S^*_{\succ}$ with the histogram $((t_1^*, J_1^*), \ldots, (t_k^*, J_k), (1, *))$ satisfies $\mathbf{s} \leq \mathbf{i}$.

Now, let $\Delta_{\mathsf{T}}(w, w') = \delta$. By definition, $\Delta_{\mathsf{H}}(\mathbf{w}_S, \mathbf{w}'_S) \geq \delta$. Denote, for any $\mathbf{s}, \mathbf{e} \in \mathbb{N}^r$, $I_{\mathbf{s},\mathbf{e}} = \{\mathbf{i} \mid \mathbf{s} \leq \mathbf{i} \leq \mathbf{e}\}$. Since $S = \bigcup_{\mathbf{s} \in S^*, \mathbf{e} \in E(I)} I_{\mathbf{s},\mathbf{e}}$ and since $|S^*| \leq r \cdot 2^r$, a simple averaging argument (similar to the one in the proof of Claim 6.11) implies that there must exist $\mathbf{s} \in S^*$ and $\mathbf{e} \in E(I)$ such that $\Delta_{\mathsf{H}}(\mathbf{w}[I_{\mathbf{s},\mathbf{e}}], \mathbf{w}'[I_{\mathbf{s},\mathbf{e}}]) \geq \delta / (r \cdot 2^r \cdot |E(I)|)$. By Lemma 5.2, this implies that $\Delta_{\mathsf{S}}(\mathbf{w}_{\mathbf{e}}, \mathbf{w}'_{\leq \mathbf{e}}) \geq \delta / (H_{\mathbf{e}} \cdot r \cdot 2^r \cdot |E(I)|) - o(1)$ and the lemma follows by definition of $\Delta_{\mathsf{S}}^{\mathsf{avg}}$ and the fact that $H_{\mathbf{e}} \leq H_m^r$. □

Having both identified the structure of the set of endpoints and established a connection between tree distance and a notion of suffix distance for partial tensors, we are now prepared to address the heart of our argument: That tree distance in a word w translates into suffix distance in one of the maximal sub-cubes in its high-dimensional lifting.

Our strategy to prove this consists of two stages: First, given our division of endpoints into "layers" by their depth in the recursion (see Claim 6.8), we show that there must exist a group of adjacent layers that incurs a large (average) suffix distance from the code. We prove this via an inductive argument where we "peel off" one group of *adjacent layers* (which we formally define in Definition 6.14 below) at a time, starting from the deepest layer.

Second, we apply a different argument within each such group of layers to show that if the group gives large suffix distance, then so must one of the endpoints that compose this group of layers.

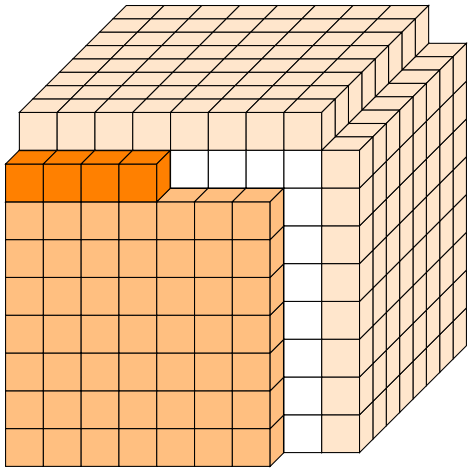

Figure 8: Endpoints of a partial tensor that corresponds to the lifting of a word of length $n = 693$. There are two groups of adjacent layers of endpoints. The first consists of the first layer, which in turn contains two endpoints: $(9, 8, 8)$ and $(8, 9, 8)$. The second consists of the second and third layers, each containing a single endpoint: $(7, 7, 9)$ and $\mathbf{n} = (4, 8, 9)$, respectively.

We refer the reader to Fig. 8 for a visual example of the endpoints in a partial tensor and their organization in adjacent groups of layers.

Let us recall our setting and fix notation for the rest of the proof. Let $n \in \mathbb{N}$ and let $\mathbf{n} = \pi^r(n)$ and $I = \{\mathbf{i} \in \mathbb{N}^r \mid \pi^{-1}(\mathbf{i}) \leq n\}$. Let $((t_1, J_1) \ldots, (t_{r'}, J_{r'}))$ be the histogram representation of $\mathbf{n}$ (see Definition 6.1). Let $w \in \Sigma^n$ be a word and let $\mathbf{w} = \mathbf{lift}^r(w)$ be its lifting to r dimensions. Let $r' \leq r$ be the size of the histogram representation of $\mathbf{n}$ and let $E_1, \ldots, E_{r'}$ be the endpoint layers as defined in Claim 6.8. For any $\ell \in [r']$, let

$$I_\ell = \bigcup_{\mathbf{e} \in E_\ell \cap E(I)} [\mathbf{e}], \qquad I_{\leq \ell} = \bigcup_{k=1}^{\ell} I_k, \qquad I_{<\ell} = I_{\leq \ell - 1}, \qquad \text{and } I_{[\ell_1, \ell_2]} = \bigcup_{\ell_1 \leq k \leq \ell_2} I_k.$$

and denote

$$\mathbf{w}_\ell = \mathbf{w}[I_{\leq \ell}], \qquad \mathbf{w}_{[\ell_1,\ell_2]} = \mathbf{w}[I_{[\ell_1,\ell_2]}], \qquad \text{and } C_\ell = C[I_{\leq \ell}], \qquad C_{[\ell_1,\ell_2]} = C[I_{[\ell_1,\ell_2]}].$$

Definition 6.14 (Adjacent Layers)**.** *A group of layers $I_{\ell_1}, \ldots, I_{\ell_2}$, for $1 \leq \ell_1 \leq \ell_2 \leq r''$, is said to be adjacent if $t_k = t_{k-1} - 1$ for all $\ell_1 < k \leq \ell_2$.*

We now proceed with the first part of our argument, namely bounding the $\Delta_{\mathsf{S}}^{\mathsf{avg}}$-distance of the any group of adjacent layers from the code. To that end, we first prove two useful statements regarding the structure of the coordinate layers. In the first claim, we characterize the endpoints of the intersection between the coordinates of a group of layers $I_{[\ell_1,\ell_2]}$ and all coordinates that were added before the ℓ_1^{th} layer (namely $I_{<\ell_1}$). Via this characterization we bound the number of these endpoints and their distance from the endpoints of $I_{[\ell_1,\ell_2]}$.

Claim 6.15. *Let $1 < \ell_1 \leq \ell_2 \leq r'$. The endpoints of the intersection $I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}$ constitutes the following disjoint union of equal-size subsets*

$$E\left(I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}\right) = \dot{\bigcup_{\substack{\ell_1 \leq k \leq \ell_2 \\ \mathbf{e} \in E_k \cap E(I)}}} \{\mathbf{e} - \mathbf{1}_j \mid j \in J_{<\ell_1}\},$$

where $J_{<\ell_1} = \bigcup_{k=1}^{\ell_1-1} J_k$ *and* $\mathbf{1}_j$ *is the* j^{th} *unit vector.*

Proof. Let $\mathbf{e} = \mathbf{e}_{k,J} \in E_k$ for some $\ell_1 \le k \le \ell_2$ (see definition in Eq. (6)) and denote $Z_{\mathbf{e}} = \{\mathbf{e}'_{k,J,j} = \mathbf{e}_{k,J} - \mathbf{1}_j \mid j \in J_{<\ell_1}\}$. It is immediate that all $Z_{\mathbf{e}}$'s are of equal size and, by inspection, they are all disjoint (see Eq. (6)). It remains to show that, indeed, $\bigcup_{k,\mathbf{e}\in E_k} Z_{\mathbf{e}} = E(I_{[\ell_1,\ell_2]} \cap I_{<\ell_1})$. Since $\mathbf{e}' \le \mathbf{e}$ for any $\mathbf{e}' \in Z_{\mathbf{e}}$ then any such $\mathbf{e}'$ is in $I_{[\ell_1,\ell_2]}$. Further, notice that for $k' < \ell_1$ and $j \in J_{k'}$, it holds that $\mathbf{e}'_{k,J,j} \le \mathbf{e}_{k',J\setminus\{j\}}$ and, therefore, any such $\mathbf{e}'$ is also in $I_{k'} \subset I_{<\ell_1}$. Next, any coordinate $\mathbf{i} \in I_k \cap I_{k'}$ for $\ell_1 \le k \le \ell_2$ and $k' < \ell_1$ satisfies $\mathbf{i} \in [\mathbf{e}'_{k,J,j}]$ for some $j \in J_{k'}$ and, therefore, $\bigcup Z_{\mathbf{e}}$ covers the intersection $I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}$. Lastly, to see why any $\mathbf{e}'_{k,J,j}$ is maximal in $I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}$, observe that if we increase $\mathbf{e}'_{k,J,j}$ by 1 at any dimension we find ourselves at a point outside $I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}$: If we do that at a dimension $j \in J_k$ for $k < \ell_1$ then we are no longer in $I_{<\ell_1}$, whereas if we add 1 to any other dimension we will no longer be in $I_{[\ell_1,\ell_2]}$ (by maximality of J in our definition of E_k in Claim 6.8). □

In the next following claim, we argue that, whenever $I_{[\ell_1,\ell_2]}$ is a *maximal* group of adjacent layers, the coordinates added by this group, namely $I_{\le\ell_2} \setminus I_{<\ell_1}$, constitute a small fraction of the "flattened" suffix that starts right after the last point in the intersection $I_{\le\ell_2} \cap I_{<\ell_1}$ is reached.

Claim 6.16. *Let* $1 < \ell_1 \le \ell_2 \le r'$ *be such that* $t_{\ell_1} < t_{\ell_1-1} - 1$. *Let* $s = \max_{\mathbf{i}' \in I_{\le\ell_2} \cap I_{<\ell_1}} \pi^{-1}(\mathbf{i}')$ *be the index of the last coordinate in* $I_{\le\ell_2} \cap I_{<\ell_1}$ *to be reached by* Π^r. *Then, it holds that* $|I_{\le\ell_2} \setminus I_{<\ell_1}| \le \alpha \cdot |\{\mathbf{i} \in I_{<\ell_1} \mid \pi^{-1}(\mathbf{i}) > s\}|$ *where* $\alpha = 1/((r-1) - \sum_{k=1}^{\ell_1-2} |J_k|)$.

Proof. By the monotonicity of Π^r (Proposition 6.4), $s = \pi^{-1}(\mathbf{e}^*)$ for some $\mathbf{e}^* \in E(I_{\le\ell_2} \cap I_{<\ell_1})$. By Claim 6.15 and Proposition 6.3, $\mathbf{e}^*$ has $(t_{\ell_1-1}, J_{\ell_1-1} \setminus \{j^*\})$ at the $(\ell_1-1)^{th}$ entry in its histogram for some $j^* \in J_{\ell_1-1}$.

Denote $S = \{\mathbf{i} \in I_{<\ell_1} \mid \pi^{-1}(\mathbf{i}) > s\}$. We prove the claim by showing there exist at least $1/\alpha$ disjoint subsets of S that are of size at least $|I_{\le\ell_2} \setminus I_{<\ell_1}|$ each.

For any $J \subseteq [r] \setminus (\bigcup_{k=1}^{\ell_1-2} J_k)$ such that $|J| = |J_{\ell_1-1}|$ and $J \prec J_{\ell_1-1}$, we define

$$S_J = [\mathbf{e}_{\ell_1-1,J}] \setminus \left(\bigcup_{J' \subset J} [\mathbf{e}_{\ell_1-1,J'}] \right).$$

For any $J \subseteq [r] \setminus (\bigcup_{k=1}^{\ell_1-2} J_k)$ such that $|J| = |J_{\ell_1-1}|$ such and $J \succ J_{\ell_1-1}$, we define

$$S_J = [\mathbf{e}_{\ell_1-1,J}] \setminus \left(\bigcup_{J' \subset J} [\mathbf{e}_{\ell_1-1,J'}] \right) - \mathbf{1}_{j(J)},$$

where $j(J) \in J$ is chosen such that $J \setminus \{j(J)\} \succeq J_{\ell_1-1} \setminus \{j^*\}$. Such $j(J)$ always exists due to Postulate 6.2 and, in particular, if $J \setminus \{j(J)\} = J_{\ell_1-1} \setminus \{j^*\}$ then $\{j(J)\} \succ \{j^*\}$.

We have defined $\binom{[r]\setminus(\bigcup_{k=1}^{\ell_1-2} J_k)}{|J_{\ell_1-1}|} - 1 \ge |[r] \setminus (\bigcup_{k=1}^{\ell_1-2} J_k)| - 1 = 1/\alpha$ sets in total.

We now show that they are all disjoint subsets of S. Notice that any coordinate $\mathbf{i} \in S_J$ for $J \prec J_{\ell_1-1}$ has value $t_{\ell-1}$ at exactly $|J_{\ell_1-1}|$ locations (namely J), while any coordinate $\mathbf{i} \in S_J$ for $J \succ J_{\ell_1-1}$ has value $t_{\ell-1}$ at exactly $|J_{\ell_1-1}| - 1$ ($J \setminus \{j(J)\}$). This, and the fact that for any $J_1, J_2 \subseteq [r]$, it holds that $[\mathbf{e}_{\ell_1-1,J_1}] \cap [\mathbf{e}_{\ell_1-1,J_2}] = [\mathbf{e}_{\ell_1-1,J_1\cap J_2}]$, implies that all S_J are disjoint.

Further, any $\mathbf{i} \in S_J$ for any $J \prec J_{\ell_1-1}$ satisfies $\pi^{-1}(\mathbf{i}) > s$, by Proposition 6.3, since $J_{\ell_1-1} \setminus \{j^*\}$ is lexicographically smaller than any J of size $|J_{\ell_1-1}|$. As for $\mathbf{i} \in S_J$ when $J \succ J_{\ell-1}$, then by our

choice of $j(J)$, we know $J \setminus \{j(J)\} \succeq J_{\ell-1} \setminus \{j^*\}$. If $J \setminus \{j(J)\} \succ J_{\ell-1} \setminus \{j^*\}$ then $\pi^{-1}(\mathbf{i}) > s$ immediately (again by Proposition 6.3). Otherwise, the ℓ_1^{th} entry in the histogram of $\mathbf{i}$, which is $(t_{\ell_1-1}-1, J')$ for some $J' \supseteq \{j(J)\}$ is lexicographically larger than the corresponding entry in $\mathbf{e}^*$, which contains $(t_{\ell_1-1}-1, \{j^*\})$ (recall the choice of $j(J)$ and the fact that $t_{\ell_1} < t_{\ell_1-1}-1$). Therefore, $\pi^{-1}(\mathbf{i}) > s$ for all $\mathbf{i} \in \bigcup S_J$. Hence, $\dot{\bigcup} S_J \subseteq S$ and $|S| \geq \sum S_J$.

Lastly, observe that $I_{\leq \ell_2} \setminus I_{<\ell_1} \subseteq [\mathbf{e}_{\ell_1-1,J_{\ell_1-1}}] \setminus (\bigcup_{J' \subset J_{\ell_1-1}} [\mathbf{e}_{\ell_1-1,J'}])$ and, therefore, by symmetry, $|I_{\leq \ell_2} \setminus I_{<\ell_1}| \leq |S_J|$ for any J such that $|J| = |J_{\ell_1-1}|$.

□

With the above two claims in hand, we prove the inductive step where we remove the deepest maximal group of adjacent layers from $I_{\leq \ell_2}$ while incurring a reasonable loss in the distance, either in the adjacent layers $I_{[\ell_1,\ell_2]}$ that we have removed, or in what remains after their removal, namely $I_{<\ell_1}$.

Lemma 6.17. *Let $1 < \ell_1 \leq \ell_2 \leq r'$ be such that $I_{[\ell_1,\ell_2]}$ is a group of adjacent layers (see Definition 6.14) and $t_{\ell_1} < t_{\ell_1} - 1$. If $\Delta_{\mathsf{T}}(w_{\ell_2}, \overline{C}_{\ell_2}) = \delta$, then either*

$$\Delta_{\mathsf{T}}\left(w_{\ell_1-1}, \overline{C}_{\ell_1-1}\right) \geq \min\left(\delta/2,\ \frac{r_{\ell_1}}{r_{\ell_1}-1} \cdot (\delta - \frac{1}{r_{\ell_1}})\right), \qquad \text{where } r_{\ell_1} = \sum_{k=\ell_1-1}^{r'} |J_k|,$$

or

$$\Delta_{\mathsf{S}}^{\mathsf{avg}}\left(\mathbf{w}_{[\ell_1,\ell_2]}, C_{[\ell_1,\ell_2]}\right) \geq \delta^*/(2r \cdot |E(I_{\leq \ell_2})|^2), \qquad \text{where } \delta^* = \min(\delta/(2H_m)^r, \delta_C).$$

Proof. Let $\mathbf{w}^*_{\ell_2} = \arg\min_{\mathbf{w} \in C_{\ell_2}} \Delta_{\mathsf{T}}(w, w_{\ell_2})$, and $\mathbf{w}^*_{\ell_1-1} = \arg\min_{\mathbf{w} \in C_{\ell_1-1}} \Delta_{\mathsf{T}}(w, w_{\ell_1-1})$. Denote by $Q = I_{[\ell_1,\ell_2]} \cap I_{<\ell_1}$ the set of coordinates in the overlap between $\mathbf{w}_{[\ell_1,\ell_2]}$ and $\mathbf{w}_{\ell_1-1}$.

If $\mathbf{w}^*_{\ell}[Q] = \mathbf{w}^*_{\ell_1-1}[Q]$ then the lemma follows immediately by Claim 6.11 (showing w is close in tree distance to the concatenation of $w^*_{\ell_1-1}$ and $w^*_{[\ell_1,\ell_2]}$) and Lemma 6.13.

Otherwise, then by the distance guaranteed by the code C and by Proposition 6.12, it holds that $\Delta_{\mathsf{S}}^{\mathsf{avg}}(\mathbf{w}^*_{[\ell_1,\ell_2]}[Q], \mathbf{w}^*_{\ell_1-1}[Q]) \geq \min_{\mathbf{e} \in E(Q)} \delta_C(\mathbf{e})/|E(Q)| \geq \min_{\mathbf{e} \in E(Q)} \delta_C(\mathbf{e})/(r \cdot |E_{[\ell_1,\ell_2]}|)$ (where $E_{[\ell_1,\ell_2]} = \bigcup_{k=\ell_1}^{\ell_2} E_k$ and the last inequality follows from the bound on $|E(Q)|$ implied by Claim 6.15). We analyze two cases:

- $\mathbf{w}^*_{\ell_1-1}[Q] = \mathbf{w}[Q]$ and, therefore, $\Delta_{\mathsf{S}}^{\mathsf{avg}}(\mathbf{w}^*_{[\ell_2,\ell_1]}[Q], \mathbf{w}[Q]) \geq \delta_C/(r \cdot |E_{[\ell_1,\ell_2]}|)$. We argue that, in this case and by the structure of $E(Q)$ shown in Claim 6.15, the disagreement over $I_{[\ell_1.\ell_2]}$ must be large as well. Let, for any $\mathbf{e} \in E_{[\ell_1,\ell_2]}$, $Z_{\mathbf{e}} = \{\mathbf{e}' \in E(Q) \mid \mathbf{e}' \leq \mathbf{e}\}$. By Claim 6.15, $Z_{\mathbf{e}} = \{\mathbf{e} - \mathbf{1}_j \mid j \in J_{<\ell_1}\}$ (where $J_{<\ell_1} = \bigcup_{k=1}^{\ell_1-1} J_k$) and, further, the $Z_{\mathbf{e}}$'s are disjoint and have equal size. Hence, short-handing $\mathbf{w}^*_{[\ell_1,\ell_2]} = \mathbf{w}^*$, we may write

  $$\Delta_{\mathsf{S}}^{\mathsf{avg}}(\mathbf{w}^*_{[\ell_1,\ell_2]}[Q], \mathbf{w}[Q]) = \mathbb{E}_{\mathbf{e} \leftarrow E(Q)}[\Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}]}, \mathbf{w}_{[\mathbf{e}]})] = \mathbb{E}_{\substack{\mathbf{e} \leftarrow E_{[\ell_1,\ell_2]} \\ \mathbf{e}' \leftarrow Z_{\mathbf{e}}}}[\Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}']}, \mathbf{w}_{[\mathbf{e}']})].$$

  By definition of Δ_{S} (Definition 5.1), we have

  $$\Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}']}, \mathbf{w}_{[\mathbf{e}']}) = \sum_{\mathbf{i} \in [\mathbf{e}']} \sigma_{\mathbf{e}'}(i) \cdot \gamma_{\mathbf{i}},$$

where $\gamma_{\mathbf{i}} = 1$ if $\mathbf{w}^*_{\mathbf{i}} \neq \mathbf{w}_{\mathbf{i}}$ and 0 otherwise. Now, since $\mathbf{e} - \mathbf{e}' = \mathbf{1}_j$ for some $j \in [r]$, then it holds that $\sigma_{\mathbf{e}}(\mathbf{i}) \geq \sigma_{\mathbf{e}'}(\mathbf{i})/2$ for any $\mathbf{i} \in [\mathbf{e}']$. This is because the distance of $\mathbf{i}$ from the reference endpoint increases by at most 1 in a single dimension, which leads to a multiplicative blow-up of at most 2 in its weight under σ (the maximal blow-up of 2 occurs at one coordinate before the last in $[\mathbf{e}]$ which becomes the last coordinate in $[\mathbf{e}']$). Hence, for $\mathbf{e}' \in Z_{\mathbf{e}}$, it holds that

$$\Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}']}, \mathbf{w}_{[\mathbf{e}']}) \leq 2 \cdot \sum_{\mathbf{i} \in [\mathbf{e}']} \sigma_{\mathbf{e}}(i) \cdot \gamma_{\mathbf{i}} \leq 2 \cdot \sum_{\mathbf{i} \in [\mathbf{e}]} \sigma_{\mathbf{e}}(i) \cdot \gamma_{\mathbf{i}} = 2 \cdot \Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}]}, \mathbf{w}_{[\mathbf{e}]}),$$

and, therefore,

$$\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}^*_{[\ell_1,\ell_2]}[Q], \mathbf{w}[Q]) \leq 2 \cdot \mathbb{E}_{\mathbf{e} \leftarrow E_{[\ell_1,\ell_2]}}[\Delta_{\mathsf{S}}(\mathbf{w}^*_{[\mathbf{e}]}, \mathbf{w}_{[\mathbf{e}]})] = 2 \cdot \Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}^*_{[\ell_1,\ell_2]}, \mathbf{w}_{[\ell_1,\ell_2]}).$$

- $\mathbf{w}^*_{\ell_1-1}[Q] \neq \mathbf{w}[Q]$. Let $\mathbf{w}' \in C_{\ell_1-1}$ be any codeword such that $\mathbf{w}'[I_{<\ell_1}] = \mathbf{w}^*_{\ell_1-1}$ (there must exist such a codeword by definition of C_{ℓ_1-1} and the fact that $\mathbf{w}^*_{\ell_1-1} \in C_{\ell_1-1}$). By the assumption in the lemma, we know that $\Delta_{\mathsf{T}}(w_{\ell_2}, w') \geq \delta$. Let i^* be the smallest integer such that $w_{\ell_2}[i^*] \neq w'[i^*]$ and let $S = \{\mathbf{i} \in I_{<\ell_1} \mid \pi^{-1}_{I_{\leq \ell_2}}(\mathbf{i}) \geq i^*\}$. Let $\tau = |I_{\leq \ell_2} \setminus Q|/(|S| + |I_{\leq \ell_2} \setminus Q|)$; This is the fraction of the "tail" of coordinates that where added at layers ℓ_1 to ℓ_2, i.e. $I_{[\ell_1,\ell_2]} \setminus I_{<\ell_1} = I_{\leq \ell_2} \setminus I_{<\ell_1}$, in the suffix of disagreements between $\mathbf{w}'$ and $\mathbf{w}_{\ell_2}$. Then, since Π^r completes its traversal over $I_{<\ell_1}$ before reaching $I_{\leq \ell_2} \setminus Q$, it holds that

$$\begin{aligned}
\Delta_{\mathsf{T}}(w_{\ell_2}, w') &= (1-\tau) \cdot \Delta_{\mathsf{H}}(\mathbf{w}'_S, \mathbf{w}_S) + \tau \cdot \Delta_{\mathsf{H}}(\mathbf{w}'_{I_{\leq \ell_2} \setminus Q}, \mathbf{w}_{I_{\leq \ell_2} \setminus Q}) \\
&\leq (1-\tau) \cdot \Delta_{\mathsf{H}}(\mathbf{w}'_S, \mathbf{w}_S) + \tau \\
&\leq (1-\tau) \cdot \Delta_{\mathsf{T}}(\mathbf{w}^*_{\ell_1-1}, \mathbf{w}_{\ell_1-1}) + \tau.
\end{aligned}$$

Lastly, by Claim 6.16, we have that $\tau \leq 1/(r - \sum_{k=1}^{\ell_1-2} |J_k|) = 1/r_{\ell_1}$ and, therefore, the proof of the lemma is complete.

□

Looking ahead, using the above lemma we peel off one group of adjacent layers at a time from our partial tensor, while preserving the distance to some extent either over the group or all other layers. At the end of this decomposition process, we are left with separate groups of adjacent layers, knowing that the distance in the partial tensor we started with is preserved in at least one of these groups. In the last step of the proof, we show that if the partial tensor is far from the code when restricted to a group of adjacent layers, then it must be far from the code over at least one maximal sub-cube contained in this group, namely at least w.r.t one of the endpoints at which we perform the test T. We prove this in the following lemma.

Lemma 6.18. *Let $1 < \ell_1 < \ell_2 \leq r'$ be such that $I_{[\ell_1,\ell_2]}$ is a group of adjacent layers (see Definition 6.14). If $\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}_{[\ell_1,\ell_2]}, C_{[\ell_1,\ell_2]}) = \delta$, then there exists $\mathbf{e} \in \bigcup_{k=\ell_1}^{\ell_2} E_k \cap E(I)$ such that $\Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, C_{[\mathbf{e}]}) \geq \min(\delta, (\ell_2 - \ell_1 + 1)^{-r} \cdot \delta_c/2)$.*

Proof. Let $\mathbf{w}^*_{\mathbf{e}} \in C_{[\mathbf{e}]}$ be the closest codeword to $\mathbf{w}$ in Δ_{S}. If, for any $\mathbf{e}_1, \mathbf{e}_2 \in \bigcup_{k=\ell_1}^{\ell_2} E_k \cap E(I)$, the words $\mathbf{w}^*_{\mathbf{e}_1}$ and $\mathbf{w}^*_{\mathbf{e}_2}$ agree on the intersection $[\mathbf{e}_1] \cap [\mathbf{e}_2]$, then we can construct a word $\mathbf{w}' \in C_{[\ell_1,\ell_2]}$ such that $\mathbf{w}'_{[\mathbf{e}]} = \mathbf{w}^*_{\mathbf{e}}$ for all $\mathbf{e}$, for which it holds that $\Delta^{\mathsf{avg}}_{\mathsf{S}}(\mathbf{w}', \mathbf{w}) = \mathbb{E}_{\mathbf{e}}[\Delta_{\mathsf{S}}(\mathbf{w}^*_{\mathbf{e}}, \mathbf{w}_{[\mathbf{e}]})]$ and, therefore, the lemma follows.

Otherwise, there must exist $\mathbf{e}_1, \mathbf{e}_2$ for which $\mathbf{w}^*_1 = \mathbf{w}^*_{\mathbf{e}_1}$ and $\mathbf{w}^*_1 = \mathbf{w}^*_{\mathbf{e}_2}$ disagree at their intersection. By the Δ_{S}-distance guarantee of the code C, this implies that the two codewords must be δ_C-far in the intersection and, therefore, one of them must be $(\delta_C/2)$-far from $\mathbf{w}$ there.

Letting $\mathbf{e}_1 = \mathbf{e}_{\ell,J}$ and $\mathbf{e}_2 = \mathbf{e}_{\ell',J'}$ (see Definition 6.7), observe that $[\mathbf{e}_1] \cap [\mathbf{e}_2] = [\mathbf{e}']$, where

$$\mathbf{e}' = ((t_1, J_1), \ldots, (t_\ell, J_\ell \cap J), \ldots, (t_{\ell'}, J_{\ell'} \cap J'), (t_{\ell'} - 1, *))$$

(assuming, w.l.o.g., $\ell < \ell'$; In the case of equality there is $J \cap J'$ at the ℓ^{th} entry and the rest of the argument follows similarly). In particular, since layers ℓ_1 to ℓ_2 are adjacent, $L_\infty(\mathbf{e}_b - \mathbf{e}') = (\ell_2 - \ell_1 + 1)$ for $b \in \{1, 2\}$. Similarly to an argument made in the proof of Lemma 6.17, this implies that $\sigma_{\mathbf{e}_b}(\mathbf{i}) \geq (\ell_2 - \ell_1 + 1)^{-r} \cdot \sigma_{\mathbf{e}'}(\mathbf{i})$ for any $\mathbf{i} \in [\mathbf{e}']$ since the distance of any such $\mathbf{i}$ from the endpoint increases by at most $(\ell_2 - \ell_1 + 1)$ in any direction (see Definition 5.1).

Now, assuming w.l.o.g. that $\Delta_{\mathsf{S}}(\mathbf{w}^*_1[[\mathbf{e}']], \mathbf{w}[[\mathbf{e}']]) > \delta_C/2$, we finish by

$$\begin{aligned}
\Delta_{\mathsf{S}}(\mathbf{w}^*_1, \mathbf{w}[[\mathbf{e}_1]]) &= \sum_{\mathbf{i} \in [\mathbf{e}_1]} \sigma_{\mathbf{e}_1}(\mathbf{i}) \cdot \gamma_{\mathbf{i}} \qquad \text{where } \gamma_{\mathbf{i}} \text{ is 1 if and only if } \mathbf{w}^*_1[\mathbf{i}] \neq \mathbf{w}[\mathbf{i}] \\
&\geq (\ell_2 - \ell_1 + 1)^{-r} \cdot \sum_{\mathbf{i} \in [\mathbf{e}']} \sigma_{\mathbf{e}'}(\mathbf{i}) \cdot \gamma_{\mathbf{i}} \\
&= (\ell_2 - \ell_1 + 1)^{-r} \cdot \Delta_{\mathsf{S}}(\mathbf{w}^*_1[[\mathbf{e}']], \mathbf{w}[[\mathbf{e}']]) > (\ell_2 - \ell_1 + 1)^{-r} \cdot \delta_C/2.
\end{aligned}$$

□

Let us now combine Lemmas 6.17 and 6.18 to complete the proof of the theorem.

We partition the layers $I_1, \ldots, I_{r'}$ into groups of maximal adjacent layers $I_{[1,\ell_1-1]}, I_{[\ell_1,\ell_2-1]}, \ldots, I_{[\ell_{r''},r']}$. By maximality, it holds that $t_{\ell_k} < t_{\ell_k - 1} - 1$ for all $k \in [r'']$. We apply Lemma 6.17 on each of these groups, starting from $I_{[\ell_{r''},r']}$. We now bound the loss in distance incurred by this inductive process. Let $r_k = \sum_{k'=\ell_k-1}^{r'} |J_{k'}|$ be as defined in Lemma 6.17 and note that $1 < r_{r''} < r_{r''-1} < \cdots < r_1 \leq r$. Define $\delta_{r''} = \delta$ and, for any $0 \leq k < r''$,

$$\delta_k = \frac{r_k}{r_k - 1} \cdot (\delta_{k+1} - \frac{1}{r_k}).$$

Then, it holds

$$\delta_0 = \left(\prod_{i=1}^{r''} \frac{r_k}{r_k - 1}\right) \cdot \left(\delta_{r''} - \sum_{k=1}^{r''} \frac{1}{r_k} \cdot \prod_{i=1}^{k} \frac{r_i - 1}{r_i}\right) \geq \delta_{r''} - \sum_{k=1}^{r''} \frac{1}{r_k} \cdot \prod_{i=1}^{k} \frac{r_i - 1}{r_i}.$$

By Claim A.1 proven in the appendix, we bound the above additive expression to obtain

$$\delta_0 \geq \delta_{r''} - (1 - H_r/r).$$

Hence, after r'' applications of Lemma 6.17, and of Lemma 6.13 over the last group of layers (i.e. $I_{[1,\ell_1-1]}$), we have decomposed the set of layers to groups of adjacent layers where there exists at least one group, let us denote it by $I_{[\ell^*_1,\ell^*_2]}$, and constants $c_1, c_2 > 0$, where

$$\Delta_{\mathsf{S}}(\mathbf{w}[I_{[\ell^*_1,\ell^*_2]}], C[I_{[\ell^*_1,\ell^*_2]}]) \geq \min\left(\frac{c_1}{H^r_m} \cdot (\delta - (1 - H_r/r)), c_2 \cdot \delta_C\right)$$

(recall $|E(I_{\leq \ell_2})|^2 \leq r^2 \cdot 2^{2r}$ by Eq. (7)).

We apply Lemma 6.18 to infer that there exists an endpoint $\mathbf{e}$ such that $\Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, C_{[\mathbf{e}]})$ is similarly bound (with different constants c_1, c_2). This completes the proof of the theorem given the strong local testability guaranteed by the test $T_{\mathbf{e}}$ and the amplification from Proposition 3.5 (we obtain the desired complexity by the facts that any endpoint $\mathbf{e}$ is in $[n^{1/r}+1, \ldots, n^{1/r}+1]$ and $H_m = O(\log n)$).

7 Locally Testable Tree Codes

In this last section, we complete our efforts and prove the main result, which we formally state below.

Theorem 7.1. *Assume there exists a linear vector tree code C over alphabet Σ with constant tree distance. Then, for any $r \geq 3$ and any $c > 0$, there exists a probabilistic binary tree code C_{ltc} satisfying the following properties:*

- *C_{ltc} has tree distance $1 - 1/n^{1-c}$,*
- *C_{ltc} is a $(\Delta_{\mathsf{T}}, \tilde{O}(n^{2/r}), 1 - H_r/r + \frac{1}{n^{1/r}})$-LTC,*
- *the n^{th} codeword symbol consists of less than $\log^{4+r}(n)$ symbols in Σ.*

In what follows, we present the last piece missing for proving Theorem 7.1. Then, in Section 7.2, we connect everything together and derive the result stated in the theorem.

7.1 Randomized Flattening

Recall that, up to this point, we have built tensor tree codes and developed a framework through which we were able to achieve meaningful notions of distance and local testability. Further, in the previous section, we have demonstrated how one can utilize these tensor codes and flatten them to obtain a code with an online encoding function in the standard sense, namely a tree code. We have even shown that such a flattened code is locally testable w.r.t. tree distance.

Despite the substantial progress so far, one major issue still remains. The flattening of a tensor code to one dimension does not preserve its distance. In fact, it can completely destroy it in the worst case (see discussion in Section 2).

In the following theorem we show that, while distance is lost when we perform our flattening deterministically, a randomized variant of the idea actually preserves distance and even amplifies it! Additionally, the randomization can be done in a way that does not harm the local testability of the code. Note, however, that the randomized transformation yields a *probabilistic* tree code (see Definition 1.3).

Theorem 7.2 (Randomized Flattening)**.** *Let $r \in \mathbb{N}$, $s : \mathbb{N} \to \mathbb{N}$ and let $c, \tau > 0$ be any constants. Assume there exists an r-dimensional tree code $C = \{C_n : \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}\}$ that has Δ_{T}-distance δ at any line (see Definition 4.6). Let $\overline{C}$ be the flattening of the code from Construction 6.5. Then, for any $\delta < 1$, there exists a (one-dimensional) probabilistic tree code $C_{\mathsf{R}} = \{C_{\mathsf{R},n} : \Sigma_{\mathsf{in}}^n \times R \to \Sigma'_{\mathsf{out}}\}$ that satisfies the following properties*

- *C_{R} has tree distance $1 - 1/n^{1-c}$.*

– *If $\overline{C}$ is a $(\Delta_{\mathsf{T}}, q, \delta_T, \epsilon)$-LTC, then C_{R} is a $(\Delta_{\mathsf{T}}, q+s, \delta_T+\tau, \epsilon')$-LTC, where*

$$\epsilon'(n) = \min(\epsilon(n), 1 - e^{-\tau s/3H_n}).$$

– *If C is explicit, then so is C_{R}.*

Further, the n^{th} codeword symbol consists of $k(n)$ symbols in Σ_{out} and $k(n) \cdot \log n$ bits, and randomness at time n consists of $k(n) \cdot \log n$ bits, where

$$k(n) < 2\log^{3+r}(n)/r\delta^r.$$

Proof of Theorem 7.2. We prove the theorem via the following construction of a probabilistic tree code from a given r-dimensional tree code C.

Construction 7.3. *Let $r \in \mathbb{N}$ and let C be an r-dimensional tree code. For any $n \in \mathbb{N}$, let $I(n) = \{\mathbf{i} \in \mathbb{N}^r \mid \pi^{-1}(\mathbf{i}) \leq n\}$ and $E(n) = E(I(n))$. We construct the code C_{R} as follows:*

- ***Randomness:*** *Let $k(i) = \log^3(i) \cdot H_{\mathbf{i}}/\delta^r$, where $\mathbf{i} = \pi^r(i)$. The randomness that is sampled at time $i \in \mathbb{N}$, is*

  $$\rho_i = (\mathbf{p}_1, \ldots, \mathbf{p}_{k(i)}),$$

  for i.i.d. $\mathbf{p}_j \leftarrow \sigma_{\mathbf{i}}$.

- ***The Encoding Function:*** *The randomness used at time n (i.e. by $C_{\mathsf{R},n}$) consists of ρ_e for any $e \in [n]$ such $\mathbf{e} = \pi^r(e)$ is in $E(n)$. Let $P(n) = \bigcup_{\mathbf{e} \in E(n)} \rho_e$. Notice that all points in $P(n)$ must be already reached by the traversal Π^r before n is (since $E(n) \subseteq I(n)$). For any $n \in \mathbb{N}$ and $m \in \Sigma_{\mathsf{in}}^n$, letting $\mathbf{n} = \pi^r(n)$ and $\mathbf{m} = \mathbf{lift}^r(m)$, the n^{th} encoding function is defined as*

  $$C_{\mathsf{R},n}(m; P(n)) = (\rho_n,\ C_{\mathbf{n}}(\mathbf{m}),\ \{(\mathbf{p}, C_{\mathbf{p}}(\mathbf{m}_{\leq \mathbf{p}}))\}_{\mathbf{p} \in P(n)}).$$

Recall the definition of endpoints $E(I)$ in Definition 6.7 and the distribution σ from Definition 5.1.

In what follows, we refer to the part of the codeword $w = C_{\mathsf{R}}(m)$ that consists of all first entries $(\rho_i)_{i \in [n]}$ as the *randomness tape* and denote it by w_R; to the part consisting of all second entries, corresponding to $(C_i(\mathbf{m}))$, as the *tensor tape* and denote it by w_C; and to the part consisting of all last entries, namely the samples from $P(n)$, as the *sample tape*, and denote it by w_P.

The construction above clearly satisfies the complexity stated in the theorem. Additionally, if C is explicit then so is C_{R}. The rest of the section is dedicated to showing C_{R} satisfies distance (as defined in Definition 1.3) and local testability (as defined in Definition 3.4).

Distance. Let m and m' be two distinct messages of length n that are the outcome of the interaction with $\mathcal{C}$ (see Fig. 1) when randomness $\rho_1, \ldots, \rho_n$ is sampled. Our goal is to lower bound the tree distance between $w = C_{\mathsf{R}}(m; \rho)$ and $w' = C_{\mathsf{R}}(m'; \rho')$, for any ρ' (rather, give a tail bound).

Let us start by bounding the distance between w and the encoding of m' under the same randomness ρ, namely $w^* = C(m'; \rho)$. Denote $\mathbf{w} = \mathbf{lift}^r(w_C)$ and $\mathbf{w}^* = \mathbf{lift}^r(w_C^*)$. Let $i^* \in [n]$ be the smallest such that $m_{i^*} \neq m'_{i^*}$ and let $\mathbf{i}^* = \pi^r(i^*)$. Denote $S = \{i \in [n] \mid \max(i^*, n^c) \leq i \leq n\}$. By inspection, it holds that

$$\Delta_{\mathsf{T}}(w, w^*) \geq \left(1 - \frac{1}{n^{1-c}}\right) \cdot \Delta_{\mathsf{H}}(w_S, w_S^*) \geq \Delta_{\mathsf{H}}(w_S, w_S^*) - \frac{1}{n^{1-c}}. \tag{8}$$

(note that the gap between the distances is biggest when $i^* = 1$ and the second disagreement appears after n^c). In what follows, we aim to bound $\Delta_{\mathsf{H}}(w_S, w^*_S)$.

Fix some $i \in S$. By the definition of $E(i)$, there exists $\mathbf{e} \in E(i)$ such that $\mathbf{i}^* \in [\mathbf{e}]$. By Lemma 5.3, since w_C and w^*_C encode messages that diverge starting i^*, we know that for such $\mathbf{e}$ it holds that $\Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, \mathbf{w}^*_{[\mathbf{e}]}) \geq \delta^r/H_{\mathbf{e}} - o(1)$. Denote $\delta(\mathbf{e}) = \delta^r/H_{\mathbf{e}} - o(1)$. It follows that

$$\mathbb{E}_{\mathbf{p}\leftarrow\sigma_{\mathbf{e}}}[\mathbf{w}_{\mathbf{p}} \neq \mathbf{w}^*_{\mathbf{p}}] = \Delta_{\mathsf{S}}(\mathbf{w}_{[\mathbf{e}]}, \mathbf{w}^*_{[\mathbf{e}]}) \geq \delta(\mathbf{e}).$$

Note that the values of $\mathbf{w}_{\mathbf{p}}, \mathbf{w}^*_{\mathbf{p}}$, for $\mathbf{p} \in \sigma_{\mathbf{e}}$, are fixed *prior* to the time $\rho_{\mathbf{e}}$ is sampled (recall Fig. 1), which happens only when reaching $\mathbf{e}$. Therefore, we can think of $\rho_{\mathbf{e}}$ as being sampled independently in $\mathbf{w}_{\mathbf{p}}, \mathbf{w}^*_{\mathbf{p}}$. Thus, letting $\gamma_{\mathbf{p}} = 1$ if $\mathbf{w}_{\mathbf{p}} \neq \mathbf{w}^*_{\mathbf{p}}$ and $\gamma_{\mathbf{p}} = 0$ otherwise, it holds that

$$\mathbb{E}_{\rho_{\mathbf{e}}}[\sum_{\mathbf{p}\in\rho_{\mathbf{e}}} \gamma_{\mathbf{p}}] = |\rho_{\mathbf{e}}| \cdot \mathbb{E}_{\mathbf{p}\leftarrow\sigma_{\mathbf{e}}}[\gamma_{\mathbf{p}}] \geq k(\pi^{-1}(\mathbf{e})) \cdot \delta(\mathbf{e}) > \log^2(\pi^{-1}(\mathbf{e})).$$

Now, due to Claim 6.8, we know that for any $\mathbf{e} \in E(i)$, it holds that $L_\infty(\mathbf{e}) = L_\infty(\mathbf{i}) = t$, for some t such that $(t-1)^r < i < t^r$. It therefore holds that $\mathbf{e}$ and $\mathbf{i}$ are both reached in the walk $\Pi^r(t)$ in the last "layer", namely after the recursive call to $\Pi^r(t-1)$. Since $\Pi^r(t)$ appends $t^r - (t-1)^r$ new coordinates after $\Pi^r(t-1)$, the number of steps it takes the traversal to reach $\mathbf{i}$ after it has reached $\mathbf{e}$ is at most $t^r - (t-1)^r < (t-1)^r/2 = i/2$. Therefore, we infer $\pi^{-1}(\mathbf{e}) > i/2$ and

$$\mathbb{E}_{\rho_{\mathbf{e}}}[\sum_{\mathbf{p}\in\rho_{\mathbf{e}}} \gamma_{\mathbf{p}}] > \log^2(i/2) > c^2 \cdot \log^2(n)/2.$$

Since $\sum \gamma_{\mathbf{p}}$ is a sum of i.i.d. random variables, we may apply the Chernoff bound to obtain

$$\Pr_{\rho_{\mathbf{e}}}\left[\sum_{\mathbf{p}\in\rho_{\mathbf{e}}} \gamma_{\mathbf{p}} \geq 1\right] > 1 - e^{-c^2 \log^2 n/6}. \tag{9}$$

Recall that a disagreement between $\mathbf{w}$ and $\mathbf{w}^*$ at some $\mathbf{p} \in \rho_{\mathbf{e}} \subseteq P(i)$ implies a disagreement between w and w^* at i in the sample tape. Therefore, Eq. (9) implies

$$\Pr_{P(i)}[w_P[i] \neq w^*_P[i]] > 1 - e^{-c^2 \log^2 n/6},$$

for any $i \in S$. Letting $D = \{i \in S \mid w_P[i] \neq w^*_P[i]\}$, we then have that $\mathbb{E}_\rho[\Delta_{\mathsf{H}}(w_P[S], w^*_P[S])] = \mathbb{E}_\rho[|D|/|S|] \geq 1 - e^{-c^2 \log^2 n/6}$ and, consequently,

$$\begin{aligned}\Pr_{\rho_1,\ldots,\rho_n}[\Delta_{\mathsf{H}}(w_P[S], w^*_P[S]) = 1] &= \Pr_{\rho_1,\ldots,\rho_n}[\Delta_{\mathsf{H}}(w_P[S], w^*_P[S]) > 1 - n^{-2}] \\ &\geq 1 - e^{-c^2 \log^2 n/6} \cdot n^2 \\ &= 1 - e^{-(c^2 \log^2 n/6) - 2\ln(n)}.\end{aligned}$$

Combining the above with Eq. (8), we have shown w is far from w^* with overwhelming probability (specifically at the sample tape), it remains to show the same from $w' = C(m'; \rho')$ for any ρ'. To that end, we argue that w and w^* disagree at a pair on their sample tape, then w disagrees at p also with any $w' \in C(m')$. To see this, observe that, by definition, the values of w_P, w^*_P and w'_P at p are of the form $(\mathbf{p}, \mathbf{w}[\mathbf{p}])$, $(\mathbf{p}, \mathbf{w}'[\mathbf{p}])$ and, resp., $(\mathbf{p}', \mathbf{w}'[\mathbf{p}'])$. Assume towards contradiction that w_P and w'_P agree. By inspection, this immediately implies w_P agrees with w^*_P as well. To conclude, any difference between w and w^* on the sample tape incurs a difference between w and any w'. This completes the argument.

Local Testability. Lastly, we show that the code C_{R} is locally testable w.r.t. tree distance assuming $\overline{C}$ is. Assume T is a q-query tester for the code $\overline{C}$. We propose a tester $T' = \{T'_n\}$ which, on input $w \in \Sigma^n$ and randomness $\rho_1, \ldots, \rho_n$, performs the steps described in Fig. 9.[17]

$T'_n(w \in \Sigma^n; \rho_1, \ldots, \rho_n)$:

1. Perform ConsistencyCheck(w) (see below).
2. Perform the test T over the word composed by all first Σ_{out}-symbols in the symbols of $\mathbf{w}$ (i.e. the ones corresponding to the encoding $C_{\mathbf{i}}(\mathbf{m}_{\leq \mathbf{i}})$ at any location i, see Construction 7.3).

ConsistencyCheck(w):

1. Sample i.i.d. $i = (i_1, \ldots, i_s)$, where $i_j \leftarrow \sigma_n$ (see Definition 5.1).
2. For any $j \in [t]$, check consistency of the pair in $w_P[i_j]$ with the randomness and tensor tapes.

Figure 9: The Tester T'

To see why the tester T' is sound, let w be such that $\Delta_{\mathsf{T}}(w, C_{\mathsf{R}}) \geq \delta_T(n) + \tau$ and let $\mathbf{w} = \mathbf{lift}^r(w)$. Let w' be the word that is obtained from w after correcting all inconsistencies. Namely, we correct all symbols in the sample tape w_P to contain the correct values taken from the randomness and tensor tapes, as expected from a codeword. By the triangle inequality, it holds that

$$\Delta_{\mathsf{T}}(w', w) + \Delta_{\mathsf{T}}(w', C_{\mathsf{R}}) \geq \delta_T + \tau.$$

Hence, either $\Delta_{\mathsf{T}}(w', C_{\mathsf{R}}) \geq \delta_T$ or $\Delta_{\mathsf{T}}(w', w) \geq \tau$. In the first case, and by the LTC guarantee of $\overline{C}$, the second step in the test T', i.e. the test T, will reject with probability at least ϵ. In what follows we argue that, in the case where w' and w are far in tree distance, the test ConsistencyCheck (defined in Fig. 9) rejects with good probability.

To see why this is the case, we rely on Lemma 5.2 to translate the tree distance between w and w' into suffix distance; Letting $\gamma_i = 1$ if $w_i \neq w'_i$ and 0 otherwise, we obtain

$$\mathbb{E}_{i \leftarrow \sigma_n}[\gamma_i] = \Delta_{\mathsf{S}}(w, w') \geq \Delta_{\mathsf{T}}(w, w')/H_n - o(1) \geq \tau/H_n - o(1).$$

Notice that ConsistencyCheck rejects when $\sum_{j=1}^{s} \gamma_{i_s} \geq 1$. Since this is a sum of i.i.d. random variables with mean $s \cdot (\tau/H_n - o(1))$, we may apply the Chernoff bound to obtain

$$\Pr_{i_1, \ldots, i_s \leftarrow \sigma_n}\left[\sum_{j=1}^{s} \gamma_{i_s} \geq 1\right] \geq 1 - e^{-\tau s/3H_n}.$$

[17] Recall that by Definition 3.4, the tester T' must output a query set Q. For simplicity, we describe a tester that either accepts or rejects its input word. The query set Q is implicit in our description and consists of any location in the input word that T' reads.

7.2 Putting it All Together

The first step for proving Theorem 7.1 is to apply an r-fold tensor over the code C to obtain C^r. By Corollary 4.8, C^r guarantees some constant tree distance δ at any line. By Corollary 5.5, C^r is a $(\Delta_{\mathsf{S}}, n^2, (2r^2 H_N \cdot \delta^r)^{-r})$-strong LTC, where $n := n(N) = L_\infty(N)$. By Theorem 6.6 and Lemma 5.3, the flattened code $\overline{C^r}$ (defined in Construction 6.5) is a $(\Delta_{\mathsf{T}}, \tilde{O}(n^{2/r}), 1 - H_r/r)$-LTC.

With the above distance and local testability properties of C^r and its flattening, our second and final step is to apply Theorem 7.2 with δ, $s = n^{2/r}$ and $\tau = 1/n^{1/r}$ to derive Theorem 7.1.

Lastly, to instantiate Theorem 7.1 and obtain a locally testable tree code, we recall the explicit tree code construction from [Sch94], specifically, a variant thereof that is presented in [Gel17, Theorem 3.1].[18]

Theorem 7.4 (Explicit Tree Codes [Sch94, Gel17])**.** *For any constant $\alpha < 1$, there exists a linear vector tree code $C = \{C_n : \{0,1\} \to \Sigma_{\mathsf{out}}\}$ with tree distance α and rate $\Theta(1/\log n)$.*

Consequently, we get the following corollary.

Corollary 7.5. *For any $r \geq 3$, there exists an explicit locally testable tree code C with rate $\Theta(1/\log^{5+r}(n))$ and distance $1 - 1/n^{1-c}$, that is $(\Delta_{\mathsf{T}}, \tilde{O}(n^{2/r}), 1 - H_r/r + n^{-1/r})$-LTC.*

Acknowledgements

We thank Irit Dinur and Salil Vadhan for insightful discussion about our model and results.

References

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120. ACM, 2013.

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[BR14] Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *IEEE Trans. Inf. Theory*, 60(11):7248–7255, 2014.

[BS04] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques,*

[18] The construction is presented in [Sch94, Gel17] as a tree code with finite depth n and a fixed output alphabet where the rate is logarithmic in n. However, one can alternatively cast their construction as an infinite-depth tree code with growing-size output alphabet, where the n^{th} codeword symbol is of size polylogarithmic in n (hence a polylogarithmic rate with constant input alphabet).

7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings, volume 3122 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2004.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 612–621. ACM, 2003.

[CHS18] Gil Cohen, Bernhard Haeupler, and Leonard J. Schulman. Explicit binary tree codes with polylogarithmic size alphabet. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 535–544. ACM, 2018.

[DEL+22] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 357–374. ACM, 2022.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[Gel17] Ran Gelles. Coding for interactive communication: A survey. *Found. Trends Theor. Comput. Sci.*, 13(1-2):1–157, 2017.

[GM12] Oded Goldreich and Or Meir. The tensor product of two good codes is not necessarily robustly testable. *Inf. Process. Lett.*, 112(8-9):351–355, 2012.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006.

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *J. ACM*, 64(2):11:1–11:42, 2017.

[Mei09] Or Meir. Combinatorial construction of locally testable codes. *SIAM J. Comput.*, 39(2):491–544, 2009.

[NPR19] Moni Naor, Omer Paneth, and Guy N. Rothblum. Incrementally verifiable computation via incremental PCPs. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 552–576. Springer, 2019.

[PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1045–1056. IEEE, 2022.

[Pud13] Pavel Pudlák. Linear tree codes and the problem of explicit constructions. *CoRR*, abs/1310.5684, 2013.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 747–756. ACM, 1993.

[Sch94] Leonard J. Schulman. Postscript from 21 september 2003 to "coding for interactive communication", 1994. Online at http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt.

[Sch96] Leonard J. Schulman. Coding for interactive communication. *IEEE Trans. Inf. Theory*, 42(6):1745–1756, 1996.

[Val05] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 472–481. Springer, 2005.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[Vid15] Michael Viderman. A combination of testability and decodability by tensor products. *Random Struct. Algorithms*, 46(3):572–598, 2015.

A Auxiliary Proofs

Claim A.1. *For any $1 < r_1 < \cdots < r_m = r$, it holds that*

$$\sum_{i=1}^{m} \frac{1}{r_i} \cdot \prod_{j=i}^{m} \frac{r_j - 1}{r_j} \leq 1 - \frac{H_r}{r}.$$

Proof of Claim A.1. Let us denote

$$T(r_1,\ldots,r_m)=\sum_{i=1}^{m}\frac{1}{r_i}\cdot\prod_{j=i}^{m}\frac{r_j-1}{r_j}.$$

We prove that $T(r_1,\ldots,r_m)\leq 1-H_r/r$ by induction on m.

For $m=1$, we have $T(r)=(r-1)/r^2<(r-1)/r$.

Assume $m>1$ and that the claim holds for any $m'<m$. We first show that T takes its maximal value when $r_i=i+1$ for any $i\in[m]$ (and $r=m+1$). To that end, we show that the derivative at any of the variables is negative.

Claim A.2. *Let $T=T(r_1,\ldots,r_m)$. Then, for any i and any $\{r_j\}_{j\neq i}$ it holds that*

$$\frac{\partial\, T}{\partial\, r_i}<0$$

when $r_i>2$.

Proof. It holds that

$$\frac{\partial\, T}{\partial\, r_i}=\frac{1}{r_i^2}\cdot\sum_{i'=1}^{i-1}\frac{1}{r_{i'}}\cdot\prod_{j=i',j\neq i}^{m}\frac{r_j-1}{r_j}+\left(\frac{2}{r_i^3}-\frac{1}{r_i^2}\right)\cdot\prod_{j=i+1}^{m}\frac{r_j-1}{r_j}.$$

Letting $X=\prod_{j=i+1}^{m}\frac{r_j-1}{r_j}>0$,

$$\frac{\partial\, T}{\partial\, r_i}=\frac{1}{r_i^2}\cdot X\cdot\sum_{i'=1}^{i-1}\frac{1}{r_{i'}}\cdot\prod_{j=i'}^{i-1}\frac{r_j-1}{r_j}+\left(\frac{2}{r_i^3}-\frac{1}{r_i^2}\right)\cdot X=\left(T(r_1,\ldots,r_{i-1})+\frac{2}{r_i}-1\right)\cdot\frac{X}{r_i^2}.$$

Now, by the inductive hypothesis we have $T(r_1,\ldots,r_{i-1})\leq 1-H_{r_{i-1}}/r_{i-1}$ and, therefore,

$$\frac{\partial\, T}{\partial\, r_i}\leq\left(2-\frac{r_i}{r_{i-1}}\cdot H_{r_{i-1}}\right)\cdot\frac{X}{r_i^3}\leq\left(2-\frac{r_{i-1}+1}{r_{i-1}}\cdot H_{r_{i-1}}\right)\cdot\frac{X}{r_i^3}$$

(recall $r_i>r_{i-1}$). The claim follows from the above since $H_x>2x/(x+1)$ for all $x>1$. □

It immediately follows by Claim A.2 that $T(r_1,\ldots,r_m)$ takes its largest value when the $r_i's$ are the smallest, namely $r_i=i+1$. In such a case, it suffices to show the following bound

$$\sum_{i=2}^{r}\frac{1}{i}\cdot\prod_{j=i}^{r}\frac{j-1}{j}\leq 1-H_r/r.$$

Observe that the product in the expression is telescopic and so it holds that

$$\prod_{j=i}^{r}\frac{j-1}{j}=\frac{i-1}{r}.$$

Hence,

$$\sum_{i=2}^{r}\frac{1}{i}\prod_{j=i}^{r}\frac{j-1}{j}=\sum_{i=2}^{r}\frac{1}{i}\cdot\left(\frac{i-1}{r}\right)=\frac{1}{r}\sum_{i=2}^{r}\left(1-\frac{1}{i}\right)=\frac{1}{r}\cdot(r-1-(H_r-1))=1-\frac{H_r}{r}.$$