Optimal Interactive Coding for Insertions, Deletions, and Substitutions

Alexander A. Sherstov and Pei Wu

***Abstract*—Interactive coding, pioneered by Schulman (FOCS ’92, STOC ’93), is concerned with making communication protocols resilient to adversarial noise. The canonical model allows the adversary to alter a small constant fraction of symbols, chosen at the adversary’s discretion, as they pass through the communication channel. Braverman, Gelles, Mao, and Ostrovsky (2015) proposed a far-reaching generalization of this model, whereby the adversary can additionally manipulate the channel by removing and inserting symbols. For any $\epsilon > 0$, they showed how to faithfully simulate any protocol in this model with corruption rate up to $1/18 - \epsilon$, using a constant-size alphabet and a constant-factor overhead in communication.**

We give an optimal simulation of any protocol in this generalized model of substitutions, insertions, and deletions, tolerating a corruption rate up to $1/4 - \epsilon$ while keeping the alphabet to a constant size and the communication overhead to a constant factor. This resolves a question due to Gelles (2015). Our corruption tolerance matches an impossibility result for corruption rate $1/4$ which holds even for substitutions alone (Braverman and Rao, STOC ’11).

***Index Terms*—Interactive coding, insertions and deletions, edit distance, tree codes, communication complexity.**

I. Introduction

CONSIDER the classical problem of transmitting a message over an unreliable channel. In its most general formulation, the problem features an omniscient and computationally unbounded adversary who controls the communication channel and can alter a small constant fraction of symbols that pass through the channel. The choice of symbols to corrupt is up to the adversary; the only guarantee is an a priori bound on the fraction of altered symbols, called the *corruption rate*. The sender’s objective is to encode the message using a somewhat longer string so as to always allow the receiver to recover the original message. This problem is the subject matter of coding theory and has been extensively studied. In particular, for any constant $\epsilon > 0$, it is known [16] how to encode an n-bit message using a string of $O(n)$ symbols from a constant-size alphabet such that the receiving party will recover the original message whenever the fraction of corrupted symbols is at most $\frac{1}{2} - \epsilon$. In seminal work, Schulman [20], [21], [22] considered a generalization of this problem to the interactive setting. Here, two parties Alice and Bob communicate back and forth according to a communication protocol agreed upon in advance. Alice and Bob privately hold inputs X and Y, respectively, which dictate their behavior throughout the communication protocol. As before, the communication channel is controlled by an adversary who can change a small constant fraction of symbols as they transit through the channel. The goal is to overcome these corruptions by cleverly simulating the original protocol with some redundant communication, as follows. The simulation leaves Alice and Bob with a record of symbols exchanged between them, where Alice’s record will generally disagree with Bob’s due to interference by the adversary. Nevertheless, they each need to be able to determine, with no further communication, the sequence of symbols that would have been exchanged in the *original* protocol on the inputs X and Y in question. Ideally, Alice and Bob’s simulation should use an alphabet of constant size and have communication cost within a constant factor of the original protocol.

A naïve solution to Schulman’s problem is for Alice and Bob to encode their individual messages with an error-correcting code developed for the noninteractive setting. This approach fails spectacularly because the adversary is only restricted by the total number of corruptions rather than the number of corruptions on a per-message basis. In particular, the adversary may choose a specific message from Alice to Bob and corrupt all symbols in it. As a result, the naïve solution cannot tolerate any corruption rate beyond $\frac{1}{m}$, where m is the total number of messages. Remarkably, Schulman [22] was able to show how to simulate any communication protocol with corruption rate up to $\frac{1}{240}$, using a constant-size alphabet and a constant-factor overhead in communication. Interactive coding has since evolved into a highly active research area with a vast literature on virtually every aspect of the problem, e.g., [19], [6], [11], [2], [8], [17], [13], [4], [3], [12], [14], [7], [10], [1], from corruption rate to communication overhead to computational complexity. We refer the reader to Gelles [9] for an up-to-date survey. Of particular interest to us is the work of Braverman and Rao [6], who proved that any communication protocol can be simulated in Schulman’s model with corruption rate up to $\frac{1}{4} - \epsilon$ for any $\epsilon > 0$, and established a matching impossibility result for corruption rate $\frac{1}{4}$. Analogous to Schulman [22], the simulation due to Braverman and Rao [6] uses a constant-size alphabet and increases the communication cost only by a constant factor.

In the canonical model discussed above, the adversary manipulates the communication channel by altering symbols. This type of manipulation is called a *substitution.* In a recent paper, Braverman, Gelles, Mao, and Ostrovsky [5] proposed a far-reaching generalization of the canonical model, whereby the adversary can additionally manipulate the channel by inserting and deleting symbols. As Braverman et al. point

A. A. Sherstov and P. Wu are with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095 USA (email: sherstov@cs.ucla.edu; pwu@cs.ucla.edu).

Manuscript received February 1, 2018; revised February 27, 2019. The first author was supported in part by NSF CAREER award CCF-1149018 and an Alfred P. Sloan Foundation Research Fellowship. The second author was supported in part by NSF CAREER award CCF-1149018. An extended abstract of this paper appeared in *Proceedings of the Fifty-Eighth IEEE Annual Symposium on Foundations of Computer Science* (FOCS), pp. 240–251, 2017.

out, insertions and deletions are considerably more difficult to handle than substitutions even in the one-way setting of coding theory. To borrow their example, Schulman and Zuckerman's polynomial-time coding and decoding algorithms [23] for insertion and deletion errors can tolerate a corruption rate of roughly $\frac{1}{100}$, in contrast to the corruption rate of $\frac{1}{2}-\epsilon$ or $\frac{1}{4}-\epsilon$ (depending on the alphabet size) achievable in the setting of substitution errors alone [16]. As their main result, Braverman et al. [5] prove that any communication protocol can be simulated in the generalized model with substitutions, insertions, and deletions as along as the corruption rate does not exceed $\frac{1}{18}-\epsilon$, for an arbitrarily small constant $\epsilon > 0$. Analogous to previous work, the simulation of Braverman et al. uses a constant-size alphabet and increases the communication cost only by a multiplicative constant.

Braverman et al. [5] and Gelles [9] posed the problem of determining the highest possible corruption rate that can be tolerated in the generalized model, and of achieving that optimal rate for every protocol. We give a detailed solution to this problem, showing that any protocol can be simulated with corruption rate up to $\frac{1}{4}-\epsilon$ for any $\epsilon > 0$. Recall that this corruption tolerance is optimal even in the setting of substitutions alone.

A. The model

Following previous work, we focus on communication protocols in *canonical form*. In such a protocol, the communication proceeds in *rounds*. The number of rounds is the same on all inputs, and each round involves Alice sending a single symbol to Bob and Bob sending a symbol back to Alice. The canonical form assumption is without loss of generality since any protocol can be brought into canonical form at the expense of doubling its communication cost.

We now describe the model of Braverman et al. [5] in more detail. Naïvely, one may be tempted to give the adversary the power to delete or insert any symbol at any time. A moment's thought reveals that such power rules out any meaningful computation. Indeed, deleting a single symbol en route from Alice to Bob will stall the communication, forcing both parties to wait on each other indefinitely to send the next symbol. Conversely, inserting a symbol into the communication channel may result in crosstalk, with both parties trying to send a symbol at the same time. Braverman et al. [5] proposed a natural and elegant formalism, to which we refer as the *BGMO model*, that avoids these abnormalities. In their model, deletions and insertions occur in pairs, with every deletion immediately followed by an insertion. In other words, the BGMO model gives the adversary the capability to intercept any symbol σ in transit from one party to the other and insert a spurious symbol σ' in its place. Crucially, the adversary is free to decide which party will receive the inserted symbol. This makes it possible for the adversary to carry out two types of attacks, illustrated in Figure 1. In a *substitution attack*, the inserted symbol is routed the same way as the original symbol. Such an attack is precisely equivalent to a substitution in Schulman's model [22]. In an *out-of-sync attack*, on the other hand, the inserted symbol is delivered to the sender of the original symbol. From the sender's point of view, an out-of-sync attack looks like a response from the other party, whereas that other party does not even know that any communication has taken place and continues to wait for an incoming symbol. Braverman et al. [5] examine a variety of candidate models, including some that are clock-driven rather than message-driven, and demonstrate that the BGMO model is essentially the only reasonable interactive formalism that allows deletions and insertions. It is important to note here that even though deletions and insertions in the BGMO model occur in pairs, the corruption pattern experienced by any given party can be an arbitrary sequence of deletions and insertions.

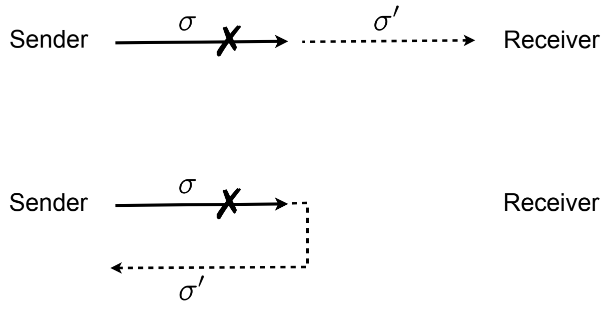

Fig. 1. A substitution attack (top) and an out-of-sync attack (bottom).

B. Our results

For the purposes of defining the corruption rate, a deletion-insertion pair in the BGMO model counts as a single corruption. This means that with corruption rate δ, the adversary is free to carry out as many as δM attacks, where M is the worst-case number of sent symbols. The main result of our paper is the following theorem, where $|\pi|$ denotes the worst-case communication cost of a protocol π.

Theorem 1: Fix an arbitrary constant $\epsilon > 0$, *and let* π *be an arbitrary protocol with alphabet* Σ. *Then there exists a simulation for* π *with alphabet size* $O(1)$ *and communication cost* $O(|\pi| \log |\Sigma|)$ *that tolerates corruption rate* $\frac{1}{4}-\epsilon$ *in the BGMO model.*

Theorem 1 matches an upper bound of $\frac{1}{4}$ on the highest possible corruption rate, due to Braverman and Rao [6], which holds even if the adversary is restricted to substitution attacks.

Theorem 1 is particularly generous in that it gives the adversary a flat budget of δM attacks, where δ is the corruption rate and M is the *maximum* number of sent symbols over all executions. Due to out-of-sync attacks, the number of symbols sent in a given execution may be substantially smaller than M. This can happen, for example, if the adversary uses out-of-sync attacks to force one of the parties to exit before his or her counterpart has reached the end of the simulation. In such case, the actual ratio of the number of attacks to the number of sent symbols may substantially exceed δ. This leads us to consider the following alternate formalism: with *normalized corruption rate* $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$, the number of substitution attacks and out-of-sync attacks in any given execution must not exceed an ϵ_{subs}

and ϵ_{oos} fraction, respectively, of the number of symbols sent in that execution. In this setting, we prove:

Theorem 2 (Normalized corruption rate): Fix an arbitrary constant $\epsilon > 0$, and let π be an arbitrary protocol with alphabet Σ. Then there exists a simulation for π with alphabet size $O(1)$ and communication cost $O(|\pi| \log |\Sigma|)$ that tolerates any normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$ in the BGMO model with

$$\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} \leqslant \frac{1}{4} - \epsilon.$$

We show that Theorem 2, too, is optimal with respect to the normalized corruption rates that it tolerates (Section V-I). In the interesting special case when the adversary is restricted to out-of-sync attacks, Theorem 2 tolerates normalized corruption rate $\frac{1}{3} - \epsilon$ for any $\epsilon > 0$. This contrasts with the maximum possible corruption rate that can be tolerated with substitutions alone, namely, $\frac{1}{4} - \epsilon$. Thus, there is a precise technical sense in which substitution attacks are more powerful than out-of-sync attacks. As we will discuss shortly, however, the mere presence of out-of-sync attacks greatly complicates the analysis and requires a fundamentally different approach.

In Theorems 1 and 2, each player computes the transcript of the simulated protocol based on his or her *entire* record of sent and received symbols, from the beginning of time until the communication stops. In Section V-H, we adapt Theorem 1 to the setting where Alice and Bob wish to know the answer by a certain round, according to each player's own counting. In particular, Braverman et al. [5] required each player to know the answer by round $(1 - 2\delta)N$, where N is the maximum number of rounds and δ is the corruption rate. With that requirement, we give a simulation that tolerates corruption rate $\frac{1}{6} - \epsilon$ for any $\epsilon > 0$, which is optimal by the impossibility result in [5, Theorem G.1].

C. Background on interactive coding

In what follows, we review relevant previous work [22], [6], [5] on interactive coding and contrast it with our approach. A key tool in this line of research is a *tree code*, a coding-theoretic primitive developed by Schulman [22]. Let Σ_{in} and Σ_{out} be nonempty finite alphabets. A tree code is any length-preserving map $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ with the property that for any input string $s \in \Sigma_{\text{in}}^*$ and any $i = 1, 2, 3, \ldots,$ the first i symbols of the codeword $C(s)$ are completely determined by the first i symbols of the input string s. A tree code has a natural representation as an infinite tree in which every vertex has arity $|\Sigma_{\text{in}}|$ and every edge is labeled with a symbol from Σ_{out}. To compute the codeword corresponding to a given input string $s = s_1 s_2 \ldots s_k$, one starts at the root and walks down the tree for k steps, choosing at the i^{th} step the branch that corresponds to s_i. The sought codeword $C(s)$, then, is the concatenation of the edge labels along this path. Tree codes are well-suited for encoding interactive communication because Alice and Bob must compute and send symbols one at a time, based on each other's responses, rather than all at once at the beginning of the protocol. In more detail, if Alice has used a tree code C to send Bob $s_1, s_2, \ldots, s_{k-1}$ and now wishes to send him s_k, she need only send the k^{th} symbol of $C(s_1 s_2 \ldots s_k)$ rather than all of $C(s_1 s_2 \ldots s_k)$. This works because by the defining properties of a tree code, the first $k-1$ symbols of $C(s_1 s_2 \ldots s_k)$ are precisely $C(s_1 s_2 \ldots s_{k-1})$ and are therefore known to Bob already. To additionally cope with adversarial substitutions, Schulman used tree codes in which different codewords are "far apart." More precisely, for any two input strings $s, s' \in \Sigma_{\text{in}}^*$ of equal length with $s_1 s_2 \ldots s_k = s'_1 s'_2 \ldots s'_k$ but $s_{k+1} \neq s'_{k+1}$, the codewords $C(s)$ and $C(s')$ disagree in a $1 - \alpha$ fraction of positions beyond the k^{th}. Schulman [22] showed the existence of such tree codes for any $\alpha > 0$, where the size of the output alphabet depends only on α and the input alphabet. Figure 2 (left) offers an illustration of the distance property for tree codes: the concatenation of the labels on the solid path should disagree with the concatenation of the labels on the dashed path in a $1-\alpha$ fraction of positions. Finally, when attempting to recover the codeword from a corrupted string $y \in \Sigma_{\text{out}}^*$, one outputs the codeword of length $|y|$ that is closest to y in Hamming distance. This recovery procedure produces the true codeword whenever y is sufficiently close to some codeword in *suffix distance*, a distance on strings that arises in a natural way from tree code properties.

We now review protocol terminology. Fix a deterministic protocol π in canonical form that Alice and Bob need to simulate on their corresponding inputs X and Y. Let Σ and n denote the alphabet and the communication cost of π, respectively. Associated to π is a tree of depth n called the *protocol tree for* π. Each vertex in this tree corresponds to the state of the protocol at some point in time, with the root corresponding to the initial state before any symbols have been exchanged, and each leaf corresponding to a final state when the communication has ended. Each internal vertex has arity $|\Sigma|$, corresponding to all possible symbols that can be transmitted at that point. Execution of π corresponds to a walk down the protocol tree, as follows. A given input X for Alice makes available precisely one outgoing edge for every internal vertex of even depth, corresponding to the symbol that she would send if the execution were to arrive at that vertex. Similarly, an input Y for Bob makes available precisely one outgoing edge for every internal vertex of odd depth. To execute π, Alice and Bob walk down the protocol tree one edge at a time, at each step selecting the edge that is dictated by the input of the player whose turn it is to speak.

We emphasize that there is no relation whatsoever between protocol trees and trees representing tree codes. They are structurally unrelated and play entirely different roles in the simulation of a protocol over an unreliable channel.

D. The Braverman–Rao simulation

We are now in a position to describe the simulation of Braverman and Rao [6] for the model with adversarial substitutions. Using the tree view of communication, we can identify Alice's input X with a set E_X of outgoing edges for the protocol tree vertices at even depths, one such edge per vertex. Analogously, Bob's input Y corresponds to a set E_Y of outgoing edges for the vertices at odd depths. Execution of π, then, corresponds to identifying the unique root-to-leaf

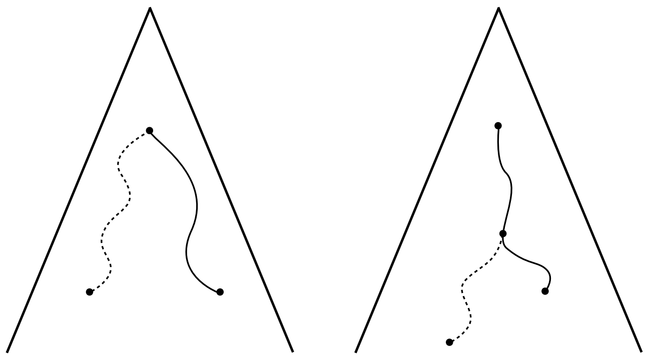

Fig. 2. Distance constraints for codewords in a tree code (left) and an edit distance tree code (right).

path made up of edges in $E_X \cup E_Y$. In Braverman and Rao's simulation, all communication is encoded and decoded using a tree code with the parameter $\alpha > 0$ set to a small constant. The simulation amounts to Alice and Bob taking turns sending each other edges from their respective sets E_X and E_Y. When it is Alice's turn to speak, she decodes the edge sequence received so far and attempts to extend the path made up of her sent and received edges by another edge from E_X, communicating this new edge to Bob. Bob acts analogously. When the communication stops, Alice decodes her complete sequence of received edges, identifies the first prefix of that sequence whose edges along with E_X contain a root-to-leaf path, and takes this root-to-leaf path to be the transcript of π on the given pair of inputs. Bob, again, acts analogously.

In the described simulation, the edge that a player sends at any given point may be irrelevant but it is never incorrect. In particular, Alice and Bob make progress in every round where they correctly decode the edge sequences that they have received so far. Braverman and Rao use a relation between suffix distance and Hamming distance to argue that with overall corruption rate $\frac{1}{4} - \epsilon$, Alice decodes her received edge sequence correctly more often than half of the time, and likewise for Bob. This means that there are a considerable number of rounds where Alice and Bob *both* decode their received sequences correctly. It follows that at some point t^*, Alice and Bob will have exchanged every edge in the root-to-leaf path in $E_X \cup E_Y$. As a final ingredient, the authors of [6] argue that the adversary's remaining budget for corruptions beyond time t^* cannot "undo" this progress, in the sense that at the end of the communication Alice and Bob will correctly decode a prefix that contains the root-to-leaf path in $E_X \cup E_Y$.

E. The BGMO simulation

We now describe the simulation of Braverman et al. [5] in the BGMO model with substitutions, insertions, and deletions. The authors of [5] draw inspiration from the classic work of Levenshtein [18], who developed codes that allow recovery from insertions and deletions in the noninteractive setting. Recall that when coding for substitution errors, one uses codewords that are far apart in Hamming distance [16]. Analogously, Levenshtein used codewords that are far apart in *edit distance*, defined for a pair of strings as the minimum number of insertions and deletions needed to transform one string into the other. To handle interactive communication, then, it is natural to start as Braverman et al. do with a tree code in which the codewords are far apart in edit distance rather than Hamming distance. The authors of [5] discover, however, that it is no longer sufficient to have distance constraints for pairs of codewords of the *same* length. Instead, for any two paths of arbitrary lengths that cross to form a lambda shape, such as the solid and dashed paths in Figure 2 (right), the associated codeword segments need to be far apart in edit distance. Braverman et al. establish the existence of such *edit distance tree codes* and develop a notion of suffix distance for them, thus providing a sufficient criterion for the recovery of the codeword from a corrupted string.

Algorithmically, the BGMO simulation departs from Braverman and Rao's in two ways. First, all communication is encoded and decoded using an edit distance tree code. Second, a different mechanism is used to decide which leaf of the protocol tree for π to output, whereby each player keeps a tally of the number of times any given leaf has been reached during the simulation and outputs the leaf with the highest tally. The resulting analysis is quite different from [6], out-of-sync attacks being the main source of difficulty. Braverman et al. start by showing that each player correctly decodes his or her received sequence of edges often enough over the course of the simulation. This does not imply progress, however. Indeed, all of Alice's correct decodings may conceivably precede all of Bob's, whereas progress is only guaranteed when the players' correct decodings are interleaved. To prove that this interleaving takes place, Braverman et al. split the simulation into n progress intervals, corresponding to the length of the longest segment recovered so far from the root-to-leaf path in $E_X \cup E_Y$. They use an amortized analysis to argue that the number of unsuccessful decodings per interval is small on the average, allowing Alice and Bob to reach the leaf on the root-to-leaf path in $E_X \cup E_Y$ at some point in the simulation. They finish the proof by arguing that the players subsequently revisit this leaf often enough that its tally outweighs that of any other leaf.

F. Our approach

There are several obstacles to improving the corruption tolerance from $\frac{1}{18} - \epsilon$ in Braverman et al. [5] to an optimal $\frac{1}{4} - \epsilon$. Some of these obstacles are of a technical nature, whereas others require a fundamental shift in approach and analysis. In the former category, we develop edit distance tree codes with stronger guarantees. Specifically, Braverman et al. use tree codes with the property that for any two paths that cross to form a lambda shape in the code tree, the edit distance between the associated codeword segments is at least a $1-\alpha$ fraction of the length of the *longer* path. We prove the existence of tree codes that guarantee a stronger lower bound on the edit distance, namely, a $1 - \alpha$ fraction of the *sum* of the lengths of the paths. This makes intuitive sense because the typical edit distance between randomly chosen strings of lengths ℓ_1 and ℓ_2 over a nontrivial alphabet is approximately $\ell_1 + \ell_2$ rather than $\max\{\ell_1, \ell_2\}$; cf. Proposition 4. Our second improvement concerns the decoding process. The notion of suffix distance used by Braverman et al. is not flexible enough to support partial recovery of a codeword. We define a more general notion that we call *k-suffix distance* and use it to give a sufficient criterion for the recovery of the first k symbols of the codeword from a corrupted string. This makes it possible to replace the tally-based output criterion of Braverman et al. with a more efficient mechanism, whereby Alice and Bob compute their output based on a *prefix* on the received edge sequence rather than the entire sequence.

The above technical improvements fall short of achieving an optimal corruption rate of $\frac{1}{4}-\epsilon$. The fundamental stumbling block is the presence of out-of-sync attacks. For one thing, Alice and Bob's transmissions can now be interleaved in a complex way, and the basic notion of a round of communication is no longer available. Out-of-sync attacks also break the symmetry between the two players in that it is now possible for one of them to receive substantially fewer symbols than the other. Finally, by directing a large number of out-of-sync attacks at one of the players, the adversary can force the simulation to stop early and thereby increase the effective error rate well beyond $\frac{1}{4}-\epsilon$. These are good reasons to doubt the existence of a simulation that tolerates corruption rate $\frac{1}{4}-\epsilon$ with substitutions, insertions, and deletions.

Our approach is nevertheless based on the intuition that out-of-sync attacks should actually *help* the analysis because they spread the brunt of a corruption between the two players rather than heaping it all on a single player. Indeed, the deletion that results from an out-of-sync attack only affects the receiver, whereas the insertion only affects the sender. This contrasts with substitution attacks, where the deletions and insertions affect exclusively the receiver. With this in mind, convexity considerations suggest that out-of-sync attacks may actually be less damaging overall than substitution attacks. To bear out this intuition, we introduce a "virtual" view of communication that centers around the *events* experienced by Alice and Bob (namely, insertions, deletions, and successful deliveries) rather than the *symbols* that they send. In this virtual view, the length of a time interval and the associated error rate are defined in terms of the number of alternations in events rather than in terms of the number of sent symbols. Among other things, the virtual view restores the symmetry between Alice and Bob and makes it impossible for the adversary to shorten the simulation using out-of-sync attacks. By way of analysis, we start by proving that corruption rate $\frac{1}{4}-\epsilon$ translates into virtual corruption rate $\frac{1}{4}-\Omega(\epsilon)$. Next, we split the simulation into n progress intervals, corresponding to the length of the longest segment recovered so far from the root-to-leaf path in $E_X \cup E_Y$, and a final interval that encompasses the remainder of the simulation. We bound the virtual length of each interval in terms of the number of corruptions and successful decodings. We then contrast this bound with the virtual length of the overall simulation, which unlike actual length is never smaller than the simulation's worst-case communication complexity. Using the previously obtained $\frac{1}{4}-\Omega(\epsilon)$ upper bound on the virtual corruption rate, we argue that Alice and Bob successfully output the root-to-leaf path in $E_X \cup E_Y$ when their communication stops.

II. Preliminaries

We start with a review of the technical preliminaries. The purpose of this section is to make the paper as self-contained as possible, and comfortably readable by a broad audience. The expert reader may wish to skim this section for notation or skip it altogether.

A. General

As usual, the complement of a set A is denoted $\overline{A}$. For arbitrary sets A and B, we define the *cardinality of A relative to B* by $|A|_B = |A \cap B|$. For a set A and a sequence s, we let $A \cup s$ denote the set of elements that occur in either A or s. We define $A \cap s$ analogously. For a logical condition C, we use the Iverson bracket:

$$\mathbf{I}[C] = \begin{cases} 1 & \text{if } C \text{ holds,} \\ 0 & \text{otherwise.} \end{cases}$$

We abbreviate $[n] = \{1, 2, \ldots, n\}$, where n is any positive integer. We let $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ and $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$ denote the set of natural numbers and the set of positive integers, respectively. We use the term *integer interval* to refer to any set of consecutive integers (finite or infinite). We perform all calculations in the extended real number system $\mathbb{R} \cup \{-\infty, \infty\}$. In particular, we have $a/0 = \infty$ for any positive number $a \in \mathbb{R}$. To simplify our notation, we further adopt the convention that

$$\frac{0}{0} = 0.$$

We let $\log x$ denote the logarithm of x to base 2. For a real-valued function $f\colon X \to \mathbb{R}$, recall that $\arg\min_{x \in X} f(x)$ denotes the set of points where f attains its minimum value. Analogously, $\arg\max_{x \in X} f(x)$ denotes the set of points where f attains its maximum value. We let $\mathrm{e} = 2.7182\ldots$ denote Euler's number. The following well-known bound [15, Proposition 1.4] is used in our proofs without further mention:

$$\sum_{i=0}^{k} \binom{n}{i} \leqslant \left(\frac{\mathrm{e}n}{k}\right)^k, \qquad k = 1, 2, \ldots, n. \tag{1}$$

B. String notation

In this manuscript, an *alphabet* Σ is any nonempty finite set of symbols other than the asterisk $*$, which we treat as a reserved symbol. Recall that Σ^* stands for the set of all strings over Σ. We let ε denote the empty string and adopt the standard shorthand $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For a nonnegative integer k, we let $\Sigma^{\leqslant k} = \{\varepsilon\} \cup \Sigma \cup \Sigma^2 \cup \cdots \cup \Sigma^k$ denote the set of all strings over Σ that have length at most k. The concatenation of the strings u and v is denoted uv. For any alphabet Σ, we let $\prec$ denote the standard partial order on Σ^* whereby $u \prec v$ if and only if $uw = v$ for a nonempty string w. The derived relations $\succ, \preceq, \succeq$ are defined as usual by

$$\begin{aligned} u \succ v \quad &\Leftrightarrow \quad v \prec u, \\ u \succeq v \quad &\Leftrightarrow \quad v \prec u \text{ or } v = u, \\ u \preceq v \quad &\Leftrightarrow \quad u \prec v \text{ or } v = u. \end{aligned}$$

A *prefix* of v is any string u with $u \preceq v$. A *suffix* of v is any string u such that $v = wu$ for some string w. A prefix or suffix of v is called *proper* if it is not equal to v. A *subsequence* of v is v itself or any string that can be obtained from v by deleting one or more symbols.

For any string v, we let $|v|$ denote the number of symbols in v. We consider the symbols of v to be indexed in the usual manner by positive integers, with v_i denoting the symbol at index i. For a set $A \subseteq \{1, 2, \ldots, |v|\}$, we use the subsequence notation $v|_A = v_{i_1} v_{i_2} \ldots v_{i_{|A|}}$, where $i_1 < i_2 < \cdots < i_{|A|}$ are the elements of A. For a number $\iota \in [0, \infty]$ in the extended real number system, we let $v_{<\iota}$ denote the substring of v obtained by keeping the symbols at indices less than ι. As special cases, we have $v_{<1} = \varepsilon$ and $v_{<\infty} = v$. The substrings $v_{\leqslant\iota}$, $v_{>\iota}$, and $v_{\geqslant\iota}$ are defined analogously. In any of these four definitions, an index range that is empty produces the empty string ε.

We view arbitrary finite sequences as strings over the corresponding alphabet. With this convention, all notational shorthands that we have introduced for strings are defined for sequences as well.

C. Edit distance

Recall that the asterisk $*$ is a reserved symbol that does not appear in any alphabet Σ in this manuscript. For a string $v \in (\Sigma \cup \{*\})^*$, we let $*(v)$ and $\overline{*}(v)$ denote the number of asterisks and non-asterisk symbols in v, respectively:

$$\begin{aligned} *(v) &= |\{i : v_i = *\}|, \\ \overline{*}(v) &= |\{i : v_i \neq *\}|. \end{aligned}$$

In particular, $*(v) + \overline{*}(v) = |v|$. We let $\not{*}(v)$ stand for the string of length $\overline{*}(v)$ obtained from v by deleting the asterisks. For example, $\not{*}(*ab*aa) = abaa$ and $\not{*}(*) = \varepsilon$ for any alphabet symbols a, b.

An *alignment* for a given pair of strings $s, r \in \Sigma^*$ is a pair of strings $S, R \in (\Sigma \cup \{*\})^*$ with the following properties:

$$\begin{aligned} &|S| = |R|, \\ &\not{*}(S) = s, \\ &\not{*}(R) = r, \\ &R_i \neq * \ \vee \ S_i \neq * && (i = 1, 2, \ldots, |S|), \\ &(R_i \neq * \ \wedge \ S_i \neq *) \implies R_i = S_i && (i = 1, 2, \ldots, |S|). \end{aligned}$$

To better distinguish alignments from ordinary strings, we reserve uppercase symbols for the former and lowercase for the latter. We write $S \parallel R$ to indicate that S and R are an alignment for some pair of strings. For an alignment $S \parallel R$, the strings $S|_A, R|_A$ for any given subset A of indices also form an alignment, to which we refer as a *subalignment* of $S \parallel R$.

The notion of a string alignment arises in an auxiliary capacity in the context of edit distance. Specifically, the *edit distance* between strings $s, r \in \Sigma^*$ is denoted $\mathrm{ED}(s, r)$ and is given by

$$\mathrm{ED}(s, r) = \min_{S \parallel R} \{*(S) + *(R)\},$$

where the minimum is over all alignments for s, r. Letting $\mathrm{LCS}(s, r)$ denote the length of the longest common subsequence of s and r, we immediately have

$$\mathrm{ED}(s, r) = |s| + |r| - 2\,\mathrm{LCS}(s, r). \tag{2}$$

The following equivalent definition is frequently useful: $\mathrm{ED}(s, r)$ is the minimum number of insertion and deletion operations necessary to transform s into r. In this equivalence, an alignment $S \parallel R$ represents a specific way to transform s into r, indicating the positions of the insertions ($S_i = *, R_i \neq *$), deletions ($S_i \neq *, R_i = *$), and unchanged symbols ($S_i = R_i \neq *$). The operational view of edit distance shows that it is a metric, with all strings s, r, t obeying

$$\mathrm{ED}(s, r) = \mathrm{ED}(r, s), \tag{3}$$
$$\mathrm{ED}(s, r) + \mathrm{ED}(r, t) \leqslant \mathrm{ED}(s, t). \tag{4}$$

Another property of edit distance is as follows.

Proposition 3: For any strings $u, v \in \Sigma^*$,

$$\mathrm{ED}(u, v) \geqslant ||u| - |v||.$$

In particular,

$$\mathrm{ED}(u, v) = ||u| - |v||$$

whenever u is a subsequence of v or vice versa.

Proof: The proposition is immediate from (2). An alternate approach is to appeal to the operational view of edit distance, as follows. An insertion or deletion changes the length of a string by at most 1. Therefore, at least $\max\{|u| - |v|, |v| - |u|\} = ||u| - |v||$ operations are needed to transform u into v. If one of the strings is a *subsequence* of the other, then either of them can clearly be transformed into the other using $||u| - |v||$ deletions or $||u| - |v||$ insertions. ■

By definition, the edit distance between a pair of strings of lengths n and m is at most $n+m$. We now show that this trivial upper bound is essentially tight when the strings are chosen uniformly at random over an alphabet of nonnegligible size.

Proposition 4: For any nonnegative integers n and m and any $0 < \alpha \leqslant 1$,

$$\mathop{\mathbf{P}}_{\substack{u \in \Sigma^n \\ v \in \Sigma^m}} [\mathrm{ED}(u, v) \leqslant (1 - \alpha)(n + m)] \leqslant \left(\frac{\mathrm{e}}{\alpha\sqrt{|\Sigma|}} \right)^{\alpha(n+m)}.$$

Proof: We may assume that

$$\frac{\mathrm{e}}{\alpha\sqrt{|\Sigma|}} \leqslant 1, \tag{5}$$

the result being trivial otherwise. Letting $\ell = \lceil \alpha(n+m)/2 \rceil$, we have

$$\begin{aligned}
\mathop{\mathbf{P}}_{\substack{u\in\Sigma^n\\ v\in\Sigma^m}}&[\mathrm{ED}(u,v) \leqslant (1-\alpha)(n+m)] \\
&= \mathop{\mathbf{P}}_{\substack{u\in\Sigma^n\\ v\in\Sigma^m}}[\mathrm{LCS}(u,v) \geqslant \ell] \\
&\leqslant \binom{n}{\ell}\binom{m}{\ell}\cdot\frac{|\Sigma|^{\ell}\cdot|\Sigma|^{n-\ell}\cdot|\Sigma|^{m-\ell}}{|\Sigma|^{n+m}} \\
&\leqslant \binom{n+m}{2\ell}\cdot\frac{1}{|\Sigma|^{\ell}} \\
&\leqslant \left(\frac{\mathrm{e}\,(n+m)}{2\ell}\cdot\frac{1}{\sqrt{|\Sigma|}}\right)^{2\ell} \\
&\leqslant \left(\frac{\mathrm{e}}{\alpha\sqrt{|\Sigma|}}\right)^{2\lceil\alpha(n+m)/2\rceil} \\
&\leqslant \left(\frac{\mathrm{e}}{\alpha\sqrt{|\Sigma|}}\right)^{\alpha(n+m)},
\end{aligned}$$

where the first, fourth, and last steps use (2), (1), and (5), respectively. ■

D. Suffix distance

We now discuss several other measures of distance for alignments and strings. For an alignment $S \parallel R$, define

$$\Delta(S,R) = \frac{*(S) + *(R)}{\overline{*}(S)}.$$

This quantity ranges in $[0,\infty]$, with the extremal values taken on. For example, $\Delta(\varepsilon,\varepsilon) = \Delta(a,a) = 0$ and $\Delta(*,a) = \infty$, where a is any alphabet symbol. The definition of Δ is motivated in large part by its relation to edit distance:

Fact 5: For any alignment $S \parallel R$ with $\Delta(S,R) < \infty$,

$$\mathrm{ED}(\not*(S), \not*(R)) \leqslant \Delta(S,R)\cdot\overline{*}(S).$$

Proof: Immediate from the definitions of ED and Δ. ■

The *suffix distance* for an alignment $S \parallel R$ is given by

$$\mathrm{SD}(S,R) = \max_{i\geqslant 1}\ \Delta(S_{\geqslant i}, R_{\geqslant i}).$$

This notion was introduced recently by Braverman et al. [5], inspired in turn by an earlier notion of suffix distance due to Schulman [22]. In our work, we must consider a more general quantity yet. Specifically, we define $\mathrm{SD}_k(S,R)$ for $0 \leqslant k \leqslant \infty$ to be the maximum $\Delta(S_{\geqslant i}, R_{\geqslant i})$ over all indices i for which $\overline{*}(S_{<i}) < k$, with the convention that $\mathrm{SD}_k(S,R) = 0$ for $k = 0$. As functions, we have

$$0 = \mathrm{SD}_0 \leqslant \mathrm{SD}_1 \leqslant \mathrm{SD}_2 \leqslant \mathrm{SD}_3 \leqslant \cdots \leqslant \mathrm{SD}_\infty = \mathrm{SD}\,. \quad (6)$$

We generalize the above definitions to strings $s, r \in \Sigma^*$ by letting

$$\mathrm{SD}(s,r) = \min_{S\parallel R}\ \mathrm{SD}(S,R), \quad (7)$$

$$\mathrm{SD}_k(s,r) = \min_{S\parallel R}\ \mathrm{SD}_k(S,R), \quad (8)$$

where in both cases the minimum is taken over all alignments $S \parallel R$ for s, r. Since there are only finitely many alignments for any pair of strings s and r, the quantities (7) and (8) can be computed in finite time.

E. Trees and tree codes

In a given tree, a *rooted path* is any path that starts at the root of the tree. The *predecessors* of a vertex v are any of the vertices on the path from the root to v, including v itself. We analogously define the *predecessors* of an edge e to be any of the edges of the rooted path that ends with e, including e itself. A *proper predecessor* of a vertex v is any predecessor of v other than v itself; analogously for edges. In keeping with standard practice, we draw trees with the root at the top and the leaves at the bottom. Accordingly, we define the *depth* of a vertex v as the length of the path from the root to v. Similarly, the *depth* of an edge e is the length of the rooted path that ends with e. We say that a given vertex v is *deeper* than another vertex u if the depth of v is larger than the depth of u; and likewise for edges.

Fix alphabets Σ_{in} and Σ_{out}. A *tree code* is any length-preserving map $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ such that the first i symbols of the output are completely determined by the first i symbols of the input. Formally,

$$\begin{aligned}
&|C(x)| = |x|, \\
&(C(x))_{\leqslant i} = C(x_{\leqslant i}), \qquad\qquad i = 0,1,2,\ldots,
\end{aligned}$$

for all $x \in \Sigma_{\text{in}}^*$. Recall that the *codewords* of C are the elements of $C(\Sigma_{\text{in}}^*)$, i.e., the strings $y \in \Sigma_{\text{out}}^*$ such that $y = C(x)$ for some x. A tree code can be represented as an infinite rooted tree in which each node has precisely $|\Sigma_{\text{in}}|$ outgoing edges, and each edge is labeled with a symbol from Σ_{out}. To compute $C(x)$ for a given string $x \in \Sigma_{\text{in}}^*$, one starts at the root and walks down the tree for $|x|$ steps, taking the edge corresponding to x_i in the i^{th} step. Then $C(x)$ is the concatenation of the $|x|$ edge labels, in the order they were encountered during the walk. If there is an a priori bound n on the length of the input string, as in this manuscript, it is sufficient to work with the restriction of the tree code to strings of length up to n. We refer to such a restriction as a *tree code of depth n*.

To allow decoding in the presence of errors, structural properties of a tree code must ensure that the encodings of distinct strings are sufficiently far apart. How this is formalized depends on the kinds of errors that must be tolerated. Previous work has considered substitution errors [22], [6] and more recently insertions and deletions [5]. We work in the latter setting and adopt structural constraints similar to those in [5].

Definition 6 (α-violation): Fix a tree code $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ and a real $0 \leqslant \alpha < 1$. A quadruple (A,B,D,E) of vertices in the tree representation of C form an *α-violation* if:

1) B is the deepest common predecessor of D and E;
2) A is any predecessor of B; and
3) $\mathrm{ED}(AD, BE) < (1-\alpha)(|AD| + |BE|)$, where $AD \in \Sigma_{\text{out}}^*$ is the concatenation of the code symbols along the path from A to D, and analogously $BE \in \Sigma_{\text{out}}^*$ is the

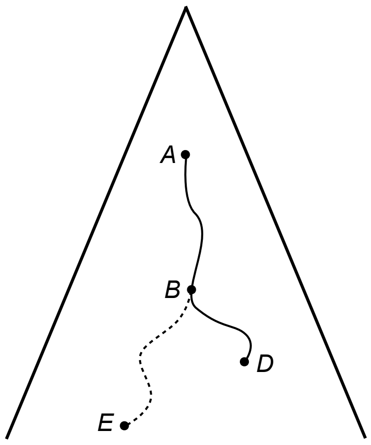

Fig. 3. A quadruple of vertices A, B, D, E involved in an α-violation.

concatenation of the code symbols along the path from B to E.

An *α-good code* is any tree code C for which no vertices A, B, D, E in its tree representation form an α-violation.

Definition 6 is illustrated in Figure 3. This definition strengthens an earlier formalism due to Braverman et al. [5], in which the inequality $\mathrm{ED}(AD, BE) < (1-\alpha)\max\{|AD|, |BE|\}$ played the role of our constraint 3). The strengthening is essential to the tight results of our paper.

Remark 7: Observe that A, B, D, E can form an α-violation for $0 \leqslant \alpha < 1$ only when

$$D \neq E,$$
$$B \neq E.$$

Indeed, suppose that one or both of these conditions fail. Then $BE = \varepsilon$ and therefore

$$\begin{aligned}\mathrm{ED}(AD, BE) &= \mathrm{ED}(AD, \varepsilon)\\ &= |AD|\\ &= |AD| + |BE|\\ &\geqslant (1-\alpha)(|AD| + |BE|),\end{aligned}$$

where the second step follows from Proposition 3.

As the next observation shows, an α-good code allows for the unique decoding of every codeword.

Fact 8: Let $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ be any α-good code, where $0 \leqslant \alpha < 1$. Then C is one-to-one.

Proof: It will be convenient to prove the contrapositive. Let $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ be a tree code such that

$$C(x') = C(x''), \tag{9}$$
$$x' \neq x'' \tag{10}$$

for some strings $x', x'' \in \Sigma_{\text{in}}^*$. Let x be the longest common prefix of x' and x''. Consider the vertices B, D, E in the tree representation of C that correspond to the input strings $x, x', x'' \in \Sigma_{\text{in}}^*$, respectively. Then

$$\mathrm{ED}(BD, BE) = 0 < (1-\alpha)(|BD| + |BE|),$$

where the first and second steps in the derivation follow from (9) and (10), respectively. Thus, the quadruple (B, B, D, E) forms an α-violation in C. ■

The following theorem, proved using the probabilistic method, ensures the existence of α-good codes with good parameters.

Theorem 9: For any alphabet Σ_{in}, any $0 < \alpha < 1$, and any integer $n \geqslant 0$, there is an α-good code $C\colon \Sigma_{\text{in}}^{\leqslant n} \to \Sigma_{\text{out}}^{\leqslant n}$ of depth n with

$$|\Sigma_{\text{out}}| = \left\lceil \frac{(10|\Sigma_{\text{in}}|)^{1/\alpha}\,\mathrm{e}}{\alpha} \right\rceil^2.$$

This theorem and its proof are adaptations of an earlier result due to Braverman et al. [5]. For the reader's convenience, we provide a complete and self-contained proof of Theorem 9 in the appendix.

F. Communication protocols

We adopt the standard two-party model of deterministic communication due to Yao [24]. In this model, Alice and Bob receive inputs $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$, respectively, where $\mathscr{X}$ and $\mathscr{Y}$ are some finite sets fixed in advance. They communicate by sending each other symbols from a fixed alphabet Σ. The most common alphabet is $\Sigma = \{0, 1\}$, but we will encounter others as well. The transfer of an alphabet symbol from one party to the other is an atomic operation to which we refer as a *transmission*. We intentionally avoid the term "message" in this paper because it is ambiguous as to the length of the content. The communication between Alice and Bob is governed by an agreed-upon *protocol* π. At any given time, the protocol specifies, based on the sequence of symbols exchanged so far between Alice and Bob, whether the communication is to continue and if so, who should send the next symbol. This next symbol is also specified by the protocol, based on the sender's input as well as the sequence of symbols exchanged so far between Alice and Bob. The *output* of the protocol π on a given pair of inputs X, Y, denoted $\pi(X, Y)$, is the complete sequence of symbols exchanged between Alice and Bob on that pair of inputs. The *communication cost* of the protocol π, denoted $|\pi|$, is the worst-case number of transmissions, or equivalently the maximum length of the protocol output on any input pair: $|\pi| = \max |\pi(X, Y)|$.

Given protocols π and Π with input space $\mathscr{X} \times \mathscr{Y}$, we say that Π *simulates* π if $\pi(X, Y) = f(\Pi(X, Y))$ for some fixed function f and all inputs $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$. To illustrate, any protocol π with alphabet Σ can be simulated in the natural manner by a protocol Π with the binary alphabet $\{0, 1\}$ and communication cost $|\Pi| \leqslant |\pi| \max\{1, \lceil \log |\Sigma| \rceil\}$. Observe that the "simulates" relation on protocols is transitive. A protocol π is said to be *in canonical form* if the following two conditions hold: (i) the number of symbols exchanged between Alice and Bob is an even integer and is the same for all inputs $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$; (ii) Alice and Bob take turns sending each other one symbol at a time, with Alice sending the first symbol. A moment's thought reveals that any protocol π can be simulated by a protocol in canonical form with the same alphabet and at most double the communication cost.

A communication protocol π over alphabet Σ can be visualized in terms of a regular tree of depth $|\pi|$, called the *protocol tree*. Every internal vertex of the protocol tree has precisely $|\Sigma|$ outgoing edges, each labeled with a distinct symbol of the alphabet. A vertex of the protocol tree corresponds in a one-to-one manner to a state of the protocol at some point in time. Specifically, the vertex reachable from the root via the path $v \in \Sigma^*$ corresponds to the point in time when the symbols exchanged between Alice and Bob so far are precisely $v_1, v_2, \ldots, v_{|v|}$, in that order. In particular, the root vertex corresponds to the point in time just before the communication starts, and a leaf corresponds to a point in time when the communication has ended. Every internal vertex of the protocol tree is said to be *owned* by either Alice or Bob, corresponding to the identity of the speaker at that point in time. For a given input $X \in \mathscr{X}$, the protocol specifies a unique outgoing edge for every vertex owned by Alice, corresponding to the symbol that she would send at that point in time with X as her input. Analogously, for any $Y \in \mathscr{Y}$, the protocol specifies a unique outgoing edge for every vertex owned by Bob. On any input pair X, Y, Alice and Bob's edges determine a unique root-to-leaf path. Execution of the protocol corresponds to a walk down this unique root-to-leaf path defined by Alice and Bob's edges, and the output of the protocol $\pi(X, Y)$ is the concatenation of the edge labels on that path. Adopting this view of communication, we will henceforth identify Alice's input with a set of edges, one for each vertex that Alice owns; and likewise for Bob. Observe that if the protocol is in canonical form, Alice and Bob's inputs are a set of outgoing edges for the even-depth vertices and a set of outgoing edges for the odd-depth vertices, respectively, one such edge per vertex.

G. The corruption model

Our review so far has focused on error-free communication. We adopt the corruption model introduced recently by Braverman et al. [5]. In this model, the communication channel between Alice and Bob is controlled by an omniscient and computationally unbounded adversary. In particular, the adversary knows Alice and Bob's protocol and their inputs. The adversary can interfere with a transmission in two different ways, illustrated in Figure 1. In a *substitution attack*, the adversary intercepts the sender's symbol σ and replaces it with a different symbol σ', which is then delivered to the receiver. In an *out-of-sync attack*, the adversary intercepts the sender's symbol σ, discards it, and then sends a spurious symbol σ' back to the sender in lieu of a response. Both a substitution attack and an out-of-sync attack involve the deletion of a symbol from the channel followed immediately by the insertion of a symbol; what makes these attacks different is how the inserted symbol is routed. On arrival, symbols manipulated by the adversary are indistinguishable from correct deliveries. As a result, Alice and Bob cannot in general tell on receipt of a transmission if it is corrupted. We remind the reader that a transmission is an atomic operation from the standpoint of interference by the adversary: either a transmission is delivered correctly and in full, or else an attack takes place and the transmission is considered to be corrupted.

Execution of a protocol is now governed not only by Alice and Bob's inputs but also by the adversary's actions. Our objective is to faithfully simulate any protocol π with only a constant-factor increase in communication cost. Our simulations will all be in canonical form, with Alice and Bob taking turns sending one symbol at a time. There are two immediate benefits to this strict alternation. First, it guarantees that the adversary cannot force *crosstalk*, with Alice and Bob attempting to send a transmission at the same time. Second, canonical form guarantees that the adversary cannot cause Alice and Bob both to *stall*, i.e., wait indefinitely on each other to send the next message. In particular, canonical form ensures that at least one of the parties is able to run the protocol to completion. The adversary may still force one of the parties to stall, e.g., by carrying out an out-of-sync attack during the next-to-last transmission. We consider an execution of the protocol to be complete as soon as the communication has stopped, due to Alice or Bob (or both) terminating.

With the adversary present, we must revisit the notion of protocol output. We define the *output* of a player in a particular execution to be the complete sequence of symbols, ordered chronologically, that that player sends and receives over the course of the execution. There is a minor technicality to address regarding which received symbols are counted toward a player's output. Due to out-of-sync attacks, Alice and Bob need not always be in agreement about how many rounds of communication they have completed. As a result, it may happen that one of the players expects the communication to continue when the other has already exited. In that case, the latter player may have one last symbol addressed to him which he or she will never retrieve from the communication channel. Since that symbol is not accessible to the player, we do not count it toward his or her input. With this minor clarification, we are prepared for formalize our notion of an interactive coding scheme.

Definition 10 (Coding scheme): Let π be a given protocol with input space $\mathscr{X} \times \mathscr{Y}$. We say that protocol Π is an *interactive coding scheme for π that tolerates corruption rate ϵ* if:

1) Π has input space $\mathscr{X} \times \mathscr{Y}$ and is in canonical form;
2) when Π is executed on a given pair of inputs $(X, Y) \in \mathscr{X} \times \mathscr{Y}$, the adversary is allowed to subject any transmission in Π to a substitution attack or out-of-sync attack, up to a total of at most $\epsilon|\Pi|$ attacks;
3) there exist functions f', f'' such that for any pair of inputs $(X, Y) \in \mathscr{X} \times \mathscr{Y}$ and any allowable behavior by the adversary, Alice's output a and Bob's output b satisfy $f'(a) = f''(b) = \pi(X, Y)$.

In this formalism, the functions f' and f'' allow Alice and Bob to interpret their respective outputs as an output of the simulated protocol π, with the requirement that these interpretations by Alice and Bob match the actual output of π on the corresponding pair of inputs.

The previous definition gives the adversary a budget of $\epsilon|\Pi|$ attacks, where $|\Pi|$ is the *maximum* length of any execution of Π. This flat budget applies even to executions that are significantly shorter than $|\Pi|$, as may happen due to out-of-sync attacks. This motivates us to define a second model, where the number of attacks in any given execution is bounded by a fraction of the *actual* length of that execution.

Definition 11 (Coding scheme with normalized corruption rate): Let π be a given protocol with input space $\mathscr{X} \times \mathscr{Y}$. We say that protocol Π is an *interactive coding scheme for π that tolerates normalized corruption rate* $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$ if:

1) Π has input space $\mathscr{X} \times \mathscr{Y}$ and is in canonical form;
2) when Π is executed on a given pair of inputs $(X, Y) \in \mathscr{X} \times \mathscr{Y}$, the adversary is allowed to subject any transmission in Π to a substitution attack or out-of-sync attack, where
   – the number of substitution attacks in any execution is at most an ϵ_{subs} fraction of the total number of transmissions in that execution, and
   – the number of out-of-sync attacks in any execution is at most an ϵ_{oos} fraction of the total number of transmissions in that execution;
3) there exist functions f', f'' such that for any pair of inputs $(X, Y) \in \mathscr{X} \times \mathscr{Y}$ and any allowable behavior by the adversary, Alice's output a and Bob's output b satisfy $f'(a) = f''(b) = \pi(X, Y)$.

In this paper, we will obtain an interactive coding scheme that achieves optimal corruption tolerance in both models (Definition 10 and 11).

III. Auxiliary results

We now prove a number of technical results on suffix distance and tree codes that are used in the design and analysis of our interactive coding schemes. Some of these results are new and some are adapted from previous work [22], [6], [5].

A. Bounds for suffix distance

Here, we collect several lower and upper bounds on suffix distance. We start with a proposition that gives bounds for alignments in terms of their subalignments.

Proposition 12: Let $S' \parallel R'$ and $S'' \parallel R''$ be given alignments. Then:

1) $\Delta(S'S'', R'R'') \leqslant \max\{\Delta(S', R'), \Delta(S'', R'')\}$;
2) $\Delta(S'S'', R'R'') \geqslant \min\{\Delta(S', R'), \Delta(S'', R'')\}$;
3) $\mathrm{SD}(S'S'', R'R'') \leqslant \max\{\mathrm{SD}(S', R'), \mathrm{SD}(S'', R'')\}$;
4) $\mathrm{SD}_k(S'S'', R'R'') \leqslant \max\{\mathrm{SD}_k(S', R'), \Delta(S'', R'')\}$ *for* $k \leqslant \overline{*}(S')$.

Proof: 1), 2) There are two cases to consider. If $\overline{*}(S') > 0$ and $\overline{*}(S'') > 0$, we have

$$\begin{aligned}\Delta(S'S'', R'R'') &= \frac{*(S'S'') + *(R'R'')}{\overline{*}(S'S'')}\\ &= \frac{*(S') + *(R')}{\overline{*}(S') + \overline{*}(S'')} + \frac{*(S'') + *(R'')}{\overline{*}(S') + \overline{*}(S'')}\\ &= \frac{\overline{*}(S')}{\overline{*}(S') + \overline{*}(S'')} \cdot \Delta(S', R')\\ &\qquad + \frac{\overline{*}(S'')}{\overline{*}(S') + \overline{*}(S'')} \cdot \Delta(S'', R'').\end{aligned}$$

In other words, $\Delta(S'S'', R'R'')$ is a weighted average of $\Delta(S', R')$ and $\Delta(S'', R'')$ and therefore lies between the minimum and maximum of these quantities.

For the complementary case, by symmetry we may assume that $\overline{*}(S') = 0$. If $S' = \varepsilon$, then $\Delta(S'S'', R'R'') = \Delta(S'', R'')$ and therefore 1) and 2) both hold. If $S' \neq \varepsilon$, then we immediately have $\Delta(S', R') = \infty$ and $\Delta(S'S'', R'R'') \geqslant \Delta(S'', R'')$, whence 1) and 2), respectively.

3) We have

$$\begin{aligned}&\mathrm{SD}(S'S'', R'R'')\\ &\quad= \max_i \Delta((S'S'')_{\geqslant i}, (R'R'')_{\geqslant i})\\ &\quad= \max\{\max_i \Delta(S''_{\geqslant i}, R''_{\geqslant i}), \max_i \Delta(S'_{\geqslant i}S'', R'_{\geqslant i}R'')\}\\ &\quad\leqslant \max\{\max_i \Delta(S''_{\geqslant i}, R''_{\geqslant i}), \max_i \Delta(S'_{\geqslant i}, R'_{\geqslant i}), \Delta(S'', R'')\}\\ &\quad= \max\{\max_i \Delta(S''_{\geqslant i}, R''_{\geqslant i}), \max_i \Delta(S'_{\geqslant i}, R'_{\geqslant i})\}\\ &\quad= \max\{\mathrm{SD}(S'', R''), \mathrm{SD}(S', R')\},\end{aligned}$$

where the third step uses 1).

4) The proof is similar to the previous item:

$$\begin{aligned}&\mathrm{SD}_k(S'S'', R'R'')\\ &\qquad= \max_i\{\Delta((S'S'')_{\geqslant i}, (R'R'')_{\geqslant i}) : \overline{*}((S'S'')_{<i}) < k\}\\ &\qquad= \max_i\{\Delta(S'_{\geqslant i}S'', R'_{\geqslant i}R'') : \overline{*}(S'_{<i}) < k\}\\ &\qquad\leqslant \max_i\{\max\{\Delta(S'_{\geqslant i}, R'_{\geqslant i}), \Delta(S'', R'')\} : \overline{*}(S'_{<i}) < k\}\\ &\qquad= \max\{\mathrm{SD}_k(S', R'), \Delta(S'', R'')\},\end{aligned}$$

where the second step is valid because $k \leqslant \overline{*}(S')$ and in particular $i \leqslant |S'|$, whereas the third step uses 1). ■

The following generic lower bound on suffix distance will also be useful.

Proposition 13: Let $k > 0$ be given. Then for all $r \in \Sigma^*$ and $s \in \Sigma^+$,

$$\mathrm{SD}_k(s, r) \geqslant 1 - \frac{|r|}{|s|}. \tag{11}$$

Proof: Fix an arbitrary alignment $S \parallel R$ for s, r. Then

$$\begin{aligned}\mathrm{SD}_k(S, R) &\geqslant \Delta(S, R)\\ &= \frac{*(S) + *(R)}{\overline{*}(S)}\\ &= \frac{*(S) + *(R)}{|s|}\\ &= \frac{*(S) + *(S) + |s| - |r|}{|s|}\\ &\geqslant \frac{|s| - |r|}{|s|},\end{aligned}$$

where the next-to-last step uses $*(S) + |s| = *(R) + |r|$. ■

B. Longest prefix decoding

In interactive coding, a sequence of symbols is encoded with a tree code and transmitted over an unreliable channel. On the receiving end, an attempt is then made to decode the sequence. The encoding and decoding are fundamentally different in that the former is fully determined by the tree code, whereas the latter allows for several reasonable approaches. In contrast to the work of Braverman et al. [5], our interactive coding schemes use *longest prefix decoding*, whereby the receiver attempts to correctly decode as long a prefix of the original sequence as possible. The following key theorem relates the length of such a prefix to the suffix distance between the original sequence and its received counterpart.

Theorem 14: Fix an α-good code $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ with $0 < \alpha < 1$. Consider a string $r \in \Sigma_{\text{out}}^*$ and codewords s', s'' of C with

$$\mathrm{SD}_k(s', r) < 1 - \alpha,$$
$$\mathrm{SD}_k(s'', r) < 1 - \alpha.$$

Then

$$s'_{\leqslant k} = s''_{\leqslant k}. \tag{12}$$

Previous work [5] settled a special case of Theorem 14 for $k = \infty$, corresponding to the correct decoding of the entire sequence. The extension to arbitrary k is essential to the optimal interactive coding schemes in our paper.

Proof of Theorem 14 (cf. Braverman et al. [5]): Let s be the longest common prefix of s' and s''. For the sake of contradiction, assume that (12) fails. Then

$$s' \neq s'', \tag{13}$$
$$|s| < k. \tag{14}$$

We will show that these two conditions force an α-violation in C, contrary to the theorem hypothesis.

Fix alignments $S' \parallel R'$ and $S'' \parallel R''$ for the string pairs s', r and s'', r, respectively, such that

$$\mathrm{SD}_k(S', R') < 1 - \alpha, \tag{15}$$
$$\mathrm{SD}_k(S'', R'') < 1 - \alpha. \tag{16}$$

Let $i', i'' \geqslant 0$ be integers with

$$s = \not{*}(S'_{\leqslant i'}), \tag{17}$$
$$s = \not{*}(S''_{\leqslant i''}). \tag{18}$$

It follows from (13) that

$$\overline{*}(S'_{>i'}) + \overline{*}(S''_{>i''}) > 0. \tag{19}$$

Observe also that r contains both $\not{*}(R'_{>i'})$ and $\not{*}(R''_{>i''})$ as suffixes, which means that one of those strings is a suffix of the other. Without loss of generality, assume that $\not{*}(R''_{>i''})$ is a suffix of $\not{*}(R'_{>i'})$ and fix an integer $j'' \geqslant 0$ such that

$$j'' \leqslant i'', \tag{20}$$
$$\not{*}(R''_{>j''}) = \not{*}(R'_{>i'}). \tag{21}$$

It follows from (14) and (17) that $\overline{*}(S'_{\leqslant i'}) < k$. Analogously, (14), (18), and (20) give $\overline{*}(S''_{\leqslant j''}) < k$. Therefore, the suffix distance bounds (15) and (16) guarantee that

$$\Delta(S'_{>i'}, R'_{>i'}) < 1 - \alpha, \tag{22}$$
$$\Delta(S''_{>j''}, R''_{>j''}) < 1 - \alpha. \tag{23}$$

In addition, (19) and (20) imply that

$$\overline{*}(S'_{>i'}) + \overline{*}(S''_{>j''}) > 0. \tag{24}$$

Now

$$\begin{aligned}
&\mathrm{ED}(\not{*}(S'_{>i'}), \not{*}(S''_{>j''})) \\
&\quad\leqslant \mathrm{ED}(\not{*}(S'_{>i'}), \not{*}(R'_{>i'})) + \mathrm{ED}(\not{*}(R'_{>i'}), \not{*}(S''_{>j''})) \\
&\quad= \mathrm{ED}(\not{*}(S'_{>i'}), \not{*}(R'_{>i'})) + \mathrm{ED}(\not{*}(S''_{>j''}), \not{*}(R'_{>i'})) \\
&\quad= \mathrm{ED}(\not{*}(S'_{>i'}), \not{*}(R'_{>i'})) + \mathrm{ED}(\not{*}(S''_{>j''}), \not{*}(R''_{>j''})) \\
&\quad\leqslant \Delta(S'_{>i'}, R'_{>i'}) \cdot \overline{*}(S'_{>i'}) + \Delta(S''_{>j''}, R''_{>j''}) \cdot \overline{*}(S''_{>j''}) \\
&\quad< (1-\alpha)(\overline{*}(S'_{>i'}) + \overline{*}(S''_{>j''})),
\end{aligned} \tag{25}$$

where the first four steps follow from (4), (3), (21), and Fact 5, respectively, and the final step is immediate from (22)–(24).

It remains to interpret our findings in terms of the tree representation of C. Let A, B, D, E be the vertices reached by following the paths $\not{*}(S''_{\leqslant j''}), s, s'', s'$, respectively, from the root of the tree. Then (25) is equivalent to $\mathrm{ED}(BE, AD) < (1-\alpha)(|BE| + |AD|)$, which is the promised α-violation. ∎

We are now in a position to describe our decoding algorithm and relate its decoding guarantees to the suffix distance between the original sequence and its received counterpart.

Theorem 15: Let $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ be an α-good code, $0 < \alpha < 1$. Then there is an algorithm $\textsc{Decode}_{C,\alpha}\colon \Sigma_{\text{out}}^* \to \Sigma_{\text{out}}^*$ that runs in finite time and obeys

$$(\textsc{Decode}_{C,\alpha}(r))_{\leqslant k} \;=\; s_{\leqslant k} \tag{26}$$

for any real $0 \leqslant k \leqslant \infty$, any codeword s, and any string $r \in \Sigma_{\text{out}}^*$ with $\mathrm{SD}_k(s, r) < 1 - \alpha$.

Proof: For a codeword s and a string r, define

$$K(s, r) = \max\{k \in \mathbb{N} \cup \{\infty\} : \mathrm{SD}_k(s, r) < 1 - \alpha\}.$$

The maximization on the right-hand side is over a nonempty set that contains $k = 0$, so that $K(s, r)$ is well-defined for every s, r pair. The algorithm is the natural one: on input r, the output of $\textsc{Decode}_{C,\alpha}$ is any $s^* \in \arg\max_s K(s, r)$, where s ranges over all codewords of C. To verify (26), let s be any codeword with $\mathrm{SD}_k(s, r) < 1 - \alpha$. Then the algorithm output s^* obeys $\mathrm{SD}_k(s^*, r) < 1 - \alpha$ and hence $s^*_{\leqslant k} = s_{\leqslant k}$ by Theorem 14.

It remains to show that $\textsc{Decode}_{C,\alpha}$ can be implemented to run in finite time. Clearly, computing $K(s, r)$ for any pair of strings s and r takes finite time. To find a codeword in $\arg\max_s K(s, r)$, it is suffices to consider codewords of length at most r/α because longer codewords s satisfy $K(s, r) = 0$ by Proposition 13. ∎

C. Frequency of good decodings

In the analysis of interactive coding schemes, one typically needs to argue that there are *many* points in time when the receiving party is able to correctly decode the sequence of symbols transmitted so far. We estimate the number of such "good decodings" using the following technical fact, closely analogous to previous work [6], [5].

Proposition 16: Fix an alignment $S \parallel R$ and define

$$G = \{i : S_i = R_i \neq *\},$$
$$D = \{i : S_i \neq *, R_i = *\},$$
$$I = \{i : S_i = *, R_i \neq *\}.$$

Algorithm 1: An algorithm for Proposition 16

```
1  $A \leftarrow \varnothing$
2  $i \leftarrow \ell$
3  while $i > 0$ do
4      if $\mathrm{SD}(S_1 S_2 \dots S_i, R_1 R_2 \dots R_i) < 1 - \alpha$ then
5          $A \leftarrow A \cup \{i\}$
6          $i \leftarrow i - 1$
7      else
8          find any index $j$ with
           $\Delta(S_j S_{j+1} \dots S_i, R_j R_{j+1} \dots R_i) \geqslant 1 - \alpha$
9          $i \leftarrow j - 1$
10     end
11 end
12 return $A$
```

Then for all $0 < \alpha < 1$,

$$|\{i \in G : \mathrm{SD}(S_1 S_2 \dots S_i, R_1 R_2 \dots R_i) < 1 - \alpha\}| \geqslant |G| - \frac{\alpha}{1-\alpha}|D| - \frac{1}{1-\alpha}|I|.$$

The notation in Proposition 16 is mnemonic, with I, D, and G denoting the positions of the inserted, deleted, and "good" (unchanged) symbols, respectively. Note that insertions and deletions play asymmetric roles in this result, insertions being more damaging than deletions.

Proof of Proposition 16 (adapted from [6], [5]): Abbreviate $\ell = |S| = |R|$ and consider Algorithm 1, which iteratively constructs a subset

$$A \subseteq \{i : \mathrm{SD}(S_1 S_2 \dots S_i, R_1 R_2 \dots R_i) < 1 - \alpha\}. \tag{27}$$

Since $\mathrm{SD}(S_1 S_2 \dots S_i, R_1 R_2 \dots R_i) \geqslant \Delta(S_i, R_i) \geqslant 1$ for every $i \in I \cup D$, we infer that $A \subseteq G$. In particular,

$$\Delta(S_{\overline{A}}, R_{\overline{A}}) = \frac{|I \cap \overline{A}| + |D \cap \overline{A}|}{|G \cap \overline{A}| + |D \cap \overline{A}|} = \frac{|I| + |D|}{|G| - |A| + |D|}. \tag{28}$$

The complementary set $\overline{A}$ is the disjoint union of the intervals $\{j, j+1, \dots, i\}$ computed by the **else** clause, each of which satisfies $\Delta(S_j S_{j+1} \dots S_i, R_j R_{j+1} \dots R_i) \geqslant 1 - \alpha$. It follows by Proposition 12, item 2) that $\Delta(S_{\overline{A}}, R_{\overline{A}}) \geqslant 1 - \alpha$, which along with (28) gives

$$|A| \geqslant |G| - \frac{\alpha}{1-\alpha}|D| - \frac{1}{1-\alpha}|I|.$$

In view of (27) and $A \subseteq G$, the proof is complete. ∎

IV. A coding scheme with a polynomial-size alphabet

We will now show how to faithfully simulate any protocol in the adversarial setting at the expense of a large increase in alphabet size and a constant-factor increase in communication cost. For an arbitrary constant $\epsilon > 0$, we give an interactive coding scheme that tolerates corruption rate $\frac{1}{4} - \epsilon$ as well as any normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$ with $\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} \leqslant \frac{1}{4} - \epsilon$. In detail, the main result of this section is as follows.

Theorem 17: Fix an arbitrary constant $\epsilon > 0$, *and let* π *be an arbitrary protocol with alphabet* Σ. *Then there exists an interactive coding scheme for* π *with alphabet size* $(|\Sigma| \cdot |\pi|)^{O(1)}$ *and communication cost* $O(|\pi|)$ *that tolerates*

1) *corruption rate* $\frac{1}{4} - \epsilon$;
2) *any normalized corruption rate* $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$ *such that* $\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} \leqslant \frac{1}{4} - \epsilon$.

As we will see later in this paper, Theorem 17 is optimal with respect to the corruption rate and normalized corruption rate that it tolerates. We have organized our proof of the theorem around nine milestones, corresponding to Sections IV-A–IV-I. Looking ahead, we will obtain the main result of this paper by improving the alphabet size to a constant.

A. The simulation

Recall from Section II-G that any protocol can be brought into canonical form at the expense of doubling its communication cost. We may therefore assume that π is in canonical form to start with. As a result, we may identify Alice's input with a set X of odd-depth edges of the protocol tree for π, and Bob's input with a set Y of even-depth edges. Execution of π corresponds to a walk down the unique root-to-leaf path in $X \cup Y$, whose length we denote by

$$n = |\pi|.$$

Analogous to previous work [6], [5], our interactive coding scheme involves Alice and Bob sending edges from their respective input sets X and Y. At any given point, Alice will send an edge e only if she has already sent every proper predecessor of e in X, and likewise for Bob. This makes it possible for the sender to represent an edge e succinctly as a pair (i, σ), where i is the index of a previous transmission by the sender that featured the grandparent of e, and $\sigma \in \Sigma \times \Sigma$ uniquely identifies e relative to that grandparent. When transmitting an edge e of depth 1 or 2, the sender sets $i = 0$ to indicate that there are no proper predecessors to refer to. Viewing each (i, σ) pair as an alphabet symbol, the resulting alphabet Σ_{in} has size at most $|\Sigma|^2$ multiplied by the total number of transmissions. The following lemma shows that given any sequence of edge representations, it is always possible to recover the corresponding sequence of edges.

Lemma 18: Consider an arbitrary point in time, and let

$$(i_1, \sigma_1), (i_2, \sigma_2), \dots (i_t, \sigma_t) \tag{29}$$

be the sequence of edge representations sent so far by one of the players. Then the sequence uniquely identifies the corresponding edges $e_1, e_2, \dots, e_t$ *sent by that player.*

Proof: The proof is by induction of t, the base case $t = 0$ being trivial. For the inductive step, let $e_1, e_2, \dots, e_{t-1}$ be the unique sequence of edges corresponding to $(i_1, \sigma_1), (i_2, \sigma_2), \dots, (i_{t-1}, \sigma_{t-1})$. Recall that $i_t \in \{0, 1, 2, \dots, t-1\}$. If $i_t \in \{1, 2, \dots, t-1\}$, then by definition (i_t, σ_t) is the grandchild of e_{i_t} that corresponds to $\sigma_t \in \Sigma \times \Sigma$. If $i_t = 0$, then by definition (i_t, σ_t) is the edge of depth 1 in Alice's case, or depth 2 in Bob's, that corresponds to σ_t. ∎

Algorithm 2: Coding scheme for Alice

```
Input: $X$ (set of Alice's edges)
1  encode and send the edge in $X$ incident to the root
2  foreach $i = 1, 2, 3, \ldots, N$ do
3      receive a symbol $r_i \in \Sigma_{\text{out}}$
4      $s \leftarrow \textsc{Decode}_{C,\alpha}(r_1 r_2 \ldots r_i)$
5      interpret $s$ as a sequence $B$ of even-depth edges
6      $\ell \leftarrow$ maximum length of a rooted path in $X \cup B$
7      compute the shortest prefix of $B$ s.t. $X \cup B$ contains
         a rooted path of length $\ell$, and let $P$ be the rooted
         path so obtained
8      $out \leftarrow$ deepest vertex in $P$
9      if $i \leqslant N - 1$ then
10         encode and send the deepest edge in $P \cap X$
             whose proper predecessors in $X$ have all been
             sent
11     end
12 end
```

Algorithm 3: Coding scheme for Bob

```
Input: $Y$ (set of Bob's edges)
1  foreach $i = 1, 2, 3, \ldots, N$ do
2      receive a symbol $r_i \in \Sigma_{\text{out}}$
3      $s \leftarrow \textsc{Decode}_{C,\alpha}(r_1 r_2 \ldots r_i)$
4      interpret $s$ as a sequence $A$ of odd-depth edges
5      $\ell \leftarrow$ maximum length of a rooted path in $Y \cup A$
6      compute the shortest prefix of $A$ s.t. $Y \cup A$ contains
         a rooted path of length $\ell$, and let $P$ be the rooted
         path so obtained
7      $out \leftarrow$ deepest vertex in $P$
8      encode and send the deepest edge in $P \cap Y$ whose
         proper predecessors in $Y$ have all been sent
9  end
```

A formal description of the interactive coding scheme is given by Algorithms 2 and 3 for Alice and Bob, respectively. In this description, $\alpha = \alpha(\epsilon) \in (0, 1)$ and $N = N(n, \epsilon)$ are parameters to be set later, and $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ is an arbitrary α-good code whose existence is ensured by Theorem 9. Alice and Bob use C to encode every transmission. In particular, the encoded symbol from Σ_{out} at any given point depends not only on the symbol from Σ_{in} being transmitted but also on the content of previous transmissions by the sender. The decoding is done using the $\textsc{Decode}_{C,\alpha}$ algorithm of Theorem 15. Apart from the initial send by Alice in line 1, the roles of two players are symmetric. In particular, the pseudocode makes it clear that Alice and Bob send at most N transmissions each. We conclude that $|\Sigma_{\text{in}}| \leqslant |\Sigma|^2 \cdot 2N$ and therefore by Theorem 9,

$$|\Sigma_{\text{out}}| = (|\Sigma| \cdot N)^{O(1/\alpha)}. \tag{30}$$

We pause to elaborate on the decoding and interpretation steps in lines 4–5 for Alice and lines 3–4 for Bob. The decoding step produces a codeword s of C, which by Fact 8 corresponds to a unique string in Σ_{in}^*. Recall that this string is of the form (29) for some integers $i_1, i_2, \ldots, i_t$ and some $\sigma_1, \sigma_2, \ldots, \sigma_t \in \Sigma \times \Sigma$. The receiving party uses the inductive procedure of Lemma 18 to convert (29) to a sequence of edges. It may happen that (29) is syntactically malformed; in that case, the receiving party interrupts the interpretation process at the longest prefix of (29) that corresponds to a legitimate sequence of edges. This completes the interpretation step, yielding a sequence of edges A for Bob and B for Alice.

In Sections IV-B–IV-I below, we examine an arbitrary but fixed execution of the interactive coding scheme. In particular, we will henceforth consider the inputs X and Y and the adversary's actions to be fixed. We allow any behavior by the adversary as long as it meets one of the criteria 1), 2) in Theorem 17. We will show that as soon as the communication stops, the variable out is set for both Alice and Bob to the leaf vertex of the unique root-to-leaf path in $X \cup Y$. This will prove Theorem 17.

B. Events

A central notion in our analysis is that of an *event*. There are three types of events: deletions, insertions, and good events. A successful transmission corresponds to a single event, which we call a *good event*. A transmission that is subject to an attack, on the other hand, corresponds to two events, namely, a *deletion event* followed immediately by an *insertion event*. Each event has an *addressee*. The addressee of a good event is defined to be the receiver of the transmission. Similarly, the deletion and insertion events that arise from a substitution attack are said to be addressed to the receiver of the transmission. In an out-of-sync attack, on the other hand, the deletion event is addressed to the intended receiver of the transmission, whereas the insertion event is addressed to the sender.

To illustrate these definitions, consider the hypothetical execution in Table I. In this example and the rest of the paper, transmissions are ordered chronologically and numbered with consecutive integers starting at 1. The columns of Table I are numbered 1 through 10, corresponding the ten transmissions sent in this execution. These ten columns are further split into subcolumns that correspond to individual events, as follows.

1) Transmissions $1, 3, 4, 5, 7, 9, 10$ result in successful deliveries, each contributing a good event addressed to the receiver of the transmission. For each of these transmissions, the entries in the sent and received rows coincide and show the symbol delivered from the sender to the receiver.
2) Transmission 2 is subject to a substitution attack, whereby the sent symbol "1" is deleted (corresponding to the "1" and $*$ entries in the sent and received rows, respectively)

TABLE I
A HYPOTHETICAL EXECUTION.

Transmission #	**1**	**2**		**3**	**4**	**5**	**6**		**7**	**8**		**9**	**10**
Addressee	B	A	A	B	A	B	A	B	A	B	A	B	A
Symbol sent	0	1	$*$	0	1	0	0	$*$	1	0	$*$	0	1
Symbol received	0	$*$	0	0	1	0	$*$	0	1	$*$	1	0	1

and a symbol of "0" is inserted in its place (corresponding to the $*$ and "0" entries in the sent and received rows, respectively). Transmission 2 thus contributes a deletion event and an insertion event, both addressed to the receiver of the transmission.

3) Transmissions 6 and 8 are subject to out-of-sync attacks, each contributing a deletion event and an insertion event. In both cases, the deletion event is addressed to the transmission's intended receiver, whereas the insertion event is addressed to the transmission's sender. In the case of transmission 6, the sent symbol "0" is deleted (corresponding to the "0" and $*$ entries in the sent and received rows, respectively) and a new symbol of "0" is spuriously sent back on behalf of the transmission's intended receiver (corresponding to the $*$ and "0" entries in the sent and received rows, respectively).

Execution of the interactive coding scheme gives rise to a string alignment $S' \parallel R'$ for Alice and a string alignment $S'' \parallel R''$ for Bob. Each position i in the strings S' and R' corresponds in a one-to-one manner to an event addressed to Alice, which is either a good event ($S'_i = R'_i$), a deletion ($S'_i \neq *, R'_i = *$), or an insertion ($S'_i = *, R'_i \neq *$). An analogous description applies to Bob's strings S'' and R''. To illustrate, the execution in Table I corresponds to

$$\begin{aligned} S' &= 1{*}101{*}1, \\ R' &= {*}01{*}111 \end{aligned}$$

and

$$\begin{aligned} S'' &= 000{*}00, \\ R'' &= 0000{*}0. \end{aligned}$$

Recall that we order transmissions chronologically and number them with consecutive integers starting at 1. For integers $i \leqslant j$, we let $S'[i,j] \parallel R'[i,j]$ denote the subalignment of $S' \parallel R'$ that corresponds to transmissions $i, i+1, \ldots, j$. Analogously, $S''[i,j] \parallel R''[i,j]$ denotes the subalignment of $S'' \parallel R''$ that corresponds to transmissions $i, i+1, \ldots, j$. We alert the reader that in our notation, S'_i and $S'[i,i]$ have completely different meanings: the former is the i^{th} symbol of S', whereas the latter is the substring of S' that corresponds to the i^{th} transmission. We define

$$\begin{aligned} G' &= \{i : S'[i,i] = R'[i,i] \neq \varepsilon\}, \\ D' &= \{i : R'[i,i] \text{ contains } *\}, \\ I' &= \{i : S'[i,i] \text{ contains } *\}. \end{aligned}$$

In words, G', D', I' are the sets of transmissions that contribute a good event, a deletion event, and an insertion event, respectively, addressed in each case to Alice. We define analogous sets for Bob:

$$\begin{aligned} G'' &= \{i : S''[i,i] = R''[i,i] \neq \varepsilon\}, \\ D'' &= \{i : R''[i,i] \text{ contains } *\}, \\ I'' &= \{i : S''[i,i] \text{ contains } *\}. \end{aligned}$$

We abbreviate

$$\begin{aligned} G &= G' \cup G'', \\ D &= D' \cup D'', \\ I &= I' \cup I''. \end{aligned}$$

We let T denote the combined number of transmissions sent by Alice and Bob. Since neither player can send more than N transmissions, we have

$$T \leqslant 2N. \tag{31}$$

The following lemma collects basic properties of the sets just introduced.

Lemma 19: The following properties hold:

1) *G' and G'' form a partition of G;*
2) *I' and I'' form a partition of I;*
3) *D' and D'' form a partition of D;*
4) *$I = D$;*
5) *$I' \setminus D' = D'' \setminus I''$;*
6) *$I'' \setminus D'' = D' \setminus I'$;*
7) *G and I form a partition of $\{1, 2, \ldots, T\}$;*
8) *G and D form a partition of $\{1, 2, \ldots, T\}$.*

Proof: Properties 1)–3) hold because any given transmission contributes at most one good event, at most one deletion event, and at most one insertion event, where each event is addressed to precisely one of the players. Property 4) holds because deletions and insertions always occur in pairs, with any given transmission generating both or neither. Property 5) follows set-theoretically from the preceding properties:

$$\begin{aligned} I' \setminus D' &= (I \setminus I'') \setminus (D \setminus D'') && \text{by 2) and 3)} \\ &= (D \setminus I'') \setminus (D \setminus D'') && \text{by 4)} \\ &= D \cap \overline{I''} \cap \overline{D \cap \overline{D''}} && \text{by Boolean algebra} \\ &= D \cap \overline{I''} \cap (\overline{D} \cup D'') && \text{by Boolean algebra} \\ &= D \cap \overline{I''} \cap D'' && \text{by Boolean algebra} \\ &= \overline{I''} \cap D'' && \text{by 3)} \\ &= D'' \setminus I'' && \text{by Boolean algebra.} \end{aligned}$$

The proof of 6) is entirely analogous. Properties 7) and 8) can be restated by saying that the transmissions can be partitioned into those that result in successful deliveries and those that are subject to an attack. ■

C. Excellent transmissions

A straightforward consequence of Lemma 18 is that the codewords $\not\ell(S'[1,t])$ and $\not\ell(S''[1,t])$ completely reveal the sequences of edges sent by Bob and by Alice, respectively, over the course of the first t transmissions. We formalize this observation below.

Lemma 20: Let $t \in \{1, 2, \ldots, T\}$ be given. Then:

1) *the string $\not\ell(S'[1,t])$ uniquely identifies the sequence of protocol tree edges that Bob sends Alice over the course of transmissions $1, 2, \ldots, t$;*
2) *the string $\not\ell(S''[1,t])$ uniquely identifies the sequence of protocol tree edges that Alice sends Bob over the course of transmissions $1, 2, \ldots, t$.*

Proof: By symmetry, it suffices to prove the former claim. By Fact 8, the codeword $\not\ell(S'[1,t]) \in \Sigma_{\text{out}}^*$ corresponds to a unique string in Σ_{in}^*, which is the sequence of edge representations that Bob sent Alice over the course of the first t transmissions. By Lemma 18, this sequence of edge representations uniquely identifies the edges themselves. ■

Of course, due to interference by the adversary, the receiving party rarely if ever has access to the exact codeword sent by his or her counterpart. This motivates us to identify sufficient conditions that allow for complete and correct decoding by the receiving party. Define

$$E' = \{i \in G' : \mathrm{SD}(S'[1,i], R'[1,i]) < 1 - \alpha\},$$
$$E'' = \{i \in G'' : \mathrm{SD}(S''[1,i], R''[1,i]) < 1 - \alpha\}.$$

We refer to E' and E'' as the sets of *excellent* transmissions for Alice and Bob, respectively. This term is borne out by the following lemma.

Lemma 21: Let $t \in \{1, 2, \ldots, T\}$ be given.

1) *If $t \in E'$, then on receipt of transmission t, Alice is able to correctly recover the complete sequence of edges that Bob has sent her by that time.*
2) *If $t \in E''$, then on receipt of transmission t, Bob is able to correctly recover the complete sequence of edges that Alice has sent him by that time.*

Proof: By symmetry, it again suffices to prove the former claim. Let $t \in E'$. Then by definition, $\mathrm{SD}(S'[1,t], R'[1,t]) < 1 - \alpha$. Taking $k = \infty$ in Theorem 15, we conclude that $\textsc{Decode}_{C,\alpha}(\not\ell(R'[1,t])) = \not\ell(S'[1,t])$. This means that on receipt of transmission t, Alice is able to correctly recover the entire codeword $\not\ell(S'[1,t])$ that Bob has sent her so far. By Lemma 20, this in turn makes it possible for Alice to correctly identify the corresponding sequence of edges. ■

D. Bad transmissions

Recall that each symbol received by Alice from the communication channel corresponds in a one-to-one manner to a good event or an insertion. Put another way, each such symbol originates in a one-to-one manner from a transmission in $G' \cup I'$. As we saw in Section IV-C, the symbols that correspond to excellent transmissions $E' \subseteq G' \cup I'$ allow Alice to correctly recover the sequence of edges that Bob has sent her so far. In all other cases, the conversion of the received string to an edge sequence can produce unpredictable results and cannot be trusted. This motivates us to define the sets of *bad* transmissions for Alice and Bob by

$$B' = (G' \cup I') \setminus E',$$
$$B'' = (G'' \cup I'') \setminus E'',$$

respectively. We abbreviate

$$B = B' \cup B''.$$

Lemma 22: The sets B' and B'' form a partition of B.

Proof:

$$\begin{aligned} B' \cap B'' &\subseteq (G' \cup I') \cap (G'' \cup I'') \\ &= (G' \cap G'') \cup (I' \cap I'') \cup (G' \cap I'') \cup (G'' \cap I') \\ &\subseteq (G' \cap G'') \cup (I' \cap I'') \cup (G \cap I) \\ &= \varnothing, \end{aligned}$$

where the last step uses items 1), 2), 7) of Lemma 19. ■

As one might expect, the number of bad transmissions is closely related to the number of attacks by the adversary. This relation is formalized by the following lemma.

Lemma 23: For any interval J with $1 \in J$,

$$|B|_J \leqslant \frac{2}{1-\alpha}|D|_J.$$

The reader will recall that $|B|_J = |B \cap J|$ and $|D|_J = |D \cap J|$ in the lemma above. We use this relative cardinality notation extensively in the rest of the paper for improved readability and ease of typesetting.

Proof of Lemma 23: Since B and D are sets of positive integers, it suffices to consider an *integer* interval $J = \{1, 2, \ldots, t\}$. Applying Proposition 16 to the alignment $S'[1,t] \parallel R'[1,t]$ shows that

$$\begin{aligned} &|E' \cap \{1,2,\ldots,t\}| \geqslant |G' \cap \{1,2,\ldots,t\}| \\ &- \frac{\alpha}{1-\alpha}|D' \cap \{1,2,\ldots,t\}| - \frac{1}{1-\alpha}|I' \cap \{1,2,\ldots,t\}|, \end{aligned}$$

which can be succinctly written as

$$|E'|_J \geqslant |G'|_J - \frac{\alpha}{1-\alpha}|D'|_J - \frac{1}{1-\alpha}|I'|_J. \tag{32}$$

Now

$$\begin{aligned} |B'|_J &= |(G' \cup I') \setminus E'|_J \\ &= |G' \cup I'|_J - |E'|_J \\ &= |G'|_J + |I'|_J - |E'|_J \\ &\leqslant \frac{\alpha}{1-\alpha}|D'|_J + \frac{2-\alpha}{1-\alpha}|I'|_J, \end{aligned} \tag{33}$$

where the first step holds by definition, the second uses the containment $E' \subseteq G'$, the third is valid by Lemma 19, item 7), and the fourth follows from (32). A symmetric argument gives

$$|B''|_J \leqslant \frac{\alpha}{1-\alpha}|D''|_J + \frac{2-\alpha}{1-\alpha}|I''|_J. \tag{34}$$

As a result,

$$\begin{aligned} |B|_J &\leqslant \frac{\alpha}{1-\alpha}(|D'|_J + |D''|_J) + \frac{2-\alpha}{1-\alpha}(|I'|_J + |I''|_J) \\ &= \frac{\alpha}{1-\alpha}|D|_J + \frac{2-\alpha}{1-\alpha}|I|_J \\ &= \frac{2}{1-\alpha}|D|_J, \end{aligned}$$

where the first step follows from (33) and (34), the second uses Lemma 19, items 2), 3), and the third uses Lemma 19, item 4). ∎

E. Virtual length

Key to our approach is a virtual view of communication that centers around *events* rather than actual transmissions. In particular, we focus on alternations in event addressee as opposed to alternations in sender. To start with, we define for an arbitrary set $Z \subseteq \mathbb{R}$ its *virtual length* by

$$|||Z||| = |G' \cup I' \cup D'|_Z + |G'' \cup I'' \cup D''|_Z. \qquad (35)$$

In other words, the virtual length $|||Z|||$ is the number of transmissions in Z that have an event addressed to Alice, plus the number of transmissions in Z that have an event addressed to Bob. It follows immediately that

$$|Z| \leqslant |||Z||| \leqslant 2|Z|$$

for any $Z \subseteq \{1, 2, \ldots, T\}$, and a moment's thought reveals that the lower and upper bounds can both be attained. We are of course interested only in subsets $Z \subseteq \{1, 2, \ldots, T\}$, but defining virtual length as we did above for arbitrary $Z \subseteq \mathbb{R}$ greatly simplifies the notation. We now show that in the special case when Z is an interval, the two summands in (35) differ by at most 1.

Lemma 24: For any interval J,

$$|||J||| \leqslant 2|G' \cup I' \cup D'|_J + 1, \qquad (36)$$
$$|||J||| \leqslant 2|G'' \cup I'' \cup D''|_J + 1 \qquad (37)$$

and

$$|||J||| \geqslant 2|G' \cup I' \cup D'|_J - 1, \qquad (38)$$
$$|||J||| \geqslant 2|G'' \cup I'' \cup D''|_J - 1. \qquad (39)$$

Proof: Consider arbitrary integers $i_1 < i_2$ such that

$$i_1 \in (G'' \cup D'' \cup I'') \setminus (G' \cup D' \cup I'),$$
$$i_2 \in (G'' \cup D'' \cup I'') \setminus (G' \cup D' \cup I').$$

The first equation states that transmission i_1 is sent by Alice and is not subject to an out-of-sync attack. Recall that a transmission causes a change of speaker if and only if it is not subject to an out-of-sync attack. As a result, a change of speaker from Alice to Bob happens immediately after transmission i_1. Since the later transmission i_2 is again sent by Alice, there must be an intermediate transmission j that causes a change of speaker from Bob to Alice. This implies

$$j \in (G' \cup D' \cup I') \setminus (G'' \cup D'' \cup I'').$$

The previous paragraph shows that the interval between any two distinct integers in $(G'' \cup D'' \cup I'') \setminus (G' \cup D' \cup I')$ contains at least one integer in $(G' \cup D' \cup I') \setminus (G'' \cup D'' \cup I'')$. We conclude that for any interval J,

$$|G'' \cup D'' \cup I''|_J \leqslant |G' \cup D' \cup I'|_J + 1.$$

Adding $|G' \cup D' \cup I'|_J$ to both sides of this inequality yields (36), whereas adding $|G'' \cup D'' \cup I''|_J$ to both sides yields (39). A symmetric argument settles the remaining inequalities (37) and (38). ∎

We now show that the combined virtual length of all transmissions is at least $2N$. This contrasts with the number of transmissions themselves, which can be significantly less than $2N$ due to out-of-sync attacks.

Lemma 25: The total virtual length of all transmissions satisfies

$$|||[1, T]||| \geqslant 2N.$$

Proof: For the communication to stop, one of the players needs to terminate. This happens only when that player has sent N symbols and received as many. Formulaically, this translates to

$$|G'' \cup D''| \geqslant N,$$
$$|G' \cup I'| \geqslant N$$

if Alice terminates first, and

$$|G' \cup D'| \geqslant N,$$
$$|G'' \cup I''| \geqslant N$$

if Bob terminates first. Either way,

$$\begin{aligned}|||[1, T]||| &= |G' \cup D' \cup I'| + |G'' \cup D'' \cup I''| \\ &\geqslant 2N,\end{aligned}$$

as was to be shown. ∎

Next, we relate the virtual length of any interval to the number of attacks experienced by Alice and Bob during that time.

Lemma 26: Let i, j be given integers with $i \leqslant j$. Then

$$|||[i, j]||| \leqslant \frac{4|D|_{[i,j]}}{\delta} + 1 \qquad (40)$$

for any $0 < \delta \leqslant 1$ such that

$$\max\{\Delta(S'[i, j], R'[i, j]),\ \Delta(S''[i, j], R''[i, j])\} \geqslant \delta. \qquad (41)$$

Proof: By hypothesis, $\Delta(S'[i, j], R'[i, j]) \geqslant \delta$ or $\Delta(S''[i, j], R''[i, j]) \geqslant \delta$. Without loss of generality, assume the former. Abbreviating $J = [i, j]$, we have

$$\frac{|D'|_J + |I'|_J}{|D'|_J + |G'|_J} \geqslant \delta,$$

which along with $\delta > 0$ gives

$$|D'|_J + |G'|_J \leqslant \frac{|D'|_J + |I'|_J}{\delta}. \qquad (42)$$

Now

$$\begin{aligned}\frac{\|J\|-1}{2} &\leqslant |G'\cup D'\cup I'|_J\\ &= |G'|_J + |D'\cup I'|_J\\ &= |G'|_J + |D'|_J + |I'\setminus D'|_J\\ &\leqslant \frac{|D'|_J+|I'|_J}{\delta} + |I'\setminus D'|_J\\ &\leqslant \frac{|D'|_J+|I'|_J+|I'\setminus D'|_J}{\delta}\\ &= \frac{|I'|_J+|I'\cup D'|_J}{\delta}\\ &\leqslant \frac{|I|_J+|I\cup D|_J}{\delta}\\ &= \frac{2|D|_J}{\delta},\end{aligned}$$

first step follows from Lemma 24, the second uses Lemma 19, items 7), 8); the fourth is valid by (42); the fifth uses $0<\delta\leqslant 1$; and the last step is immediate from Lemma 19, item 4). ■

Finally, we derive a useful bound on the virtual length of an interval in terms of the number of excellent and bad transmissions in it.

Lemma 27: For any interval J,

$$\|J\| \leqslant 2(|B|_J + |E'|_J) + 1, \tag{43}$$
$$\|J\| \leqslant 2(|B|_J + |E''|_J) + 1. \tag{44}$$

Proof: By symmetry, it suffices to prove (43). We have

$$\begin{aligned}D'\setminus I' &= I''\setminus D''\\ &\subseteq I''\\ &\subseteq I''\cup(G''\setminus E'')\\ &= (I''\cup G'')\setminus E''\\ &= B'',\end{aligned} \tag{45}$$

where the first and fourth steps use items 6) and 7), respectively, of Lemma 19. Now (43) can be verified as follows:

$$\begin{aligned}\frac{\|J\|-1}{2} &\leqslant |G'\cup I'\cup D'|_J\\ &= |G'\cup I'|_J + |D'\setminus(G'\cup I')|_J\\ &= |G'\cup I'|_J + |D'\setminus I'|_J\\ &= |E'|_J + |(G'\cup I')\setminus E'|_J + |D'\setminus I'|_J\\ &= |E'|_J + |B'|_J + |D'\setminus I'|_J\\ &\leqslant |E'|_J + |B'|_J + |B''|_J\\ &= |E'|_J + |B|_J,\end{aligned}$$

where the first step is valid by Lemma 36; the third step uses Lemma 19, item 8); the fourth step follows from the containment $E'\subseteq G'$; the fifth step applies the definition of B'; the sixth step is immediate from (45); and the final step is justified by Lemma 22. ■

F. Virtual corruption rate

In keeping with our focus on events rather than transmissions, we define

$$\operatorname{corr} J = \frac{|D\cap J|}{\|J\|}$$

for any interval J. We refer to this quantity as the *virtual corruption rate* of J. The idea of normalizing the corruption rate relative to execution length was previously used by Agrawal, Gelles, and Sahai [1]. Our notion of virtual corruption rate is somewhat more subtle in that it takes into account not only the execution length but also the numbers of attacks of each type. The next lemma shows that over the course of the execution, the virtual corruption rate is relatively low.

Lemma 28: Assumptions 1) and 2) in Theorem 17 imply

$$\operatorname{corr}[1,T] \leqslant \frac{1}{4}-\epsilon \tag{46}$$

and

$$\operatorname{corr}[1,T] \leqslant \frac{1}{4}-\frac{\epsilon}{2}, \tag{47}$$

respectively.

Proof: Assumption 1) states that the total number of attacks does not exceed a $\frac{1}{4}-\epsilon$ fraction of the worst-case communication cost of the interactive coding scheme. Formulaically,

$$|D| \leqslant \left(\frac{1}{4}-\epsilon\right)\cdot 2N.$$

As a result,

$$\operatorname{corr}[1,T] = \frac{|D|}{\|[1,T]\|} \leqslant \frac{1}{4}-\epsilon,$$

where the second step uses Lemma 25.

Assumption 2) states that

$$T_{\text{subs}} + \frac{3}{4}T_{\text{oos}} \leqslant \left(\frac{1}{4}-\epsilon\right)T,$$

where T_{subs} and T_{oos} denote the total number of substitution attacks and the total number of out-of-sync attacks, respectively. Straightforward manipulations now reveal that

$$\frac{T_{\text{subs}}+T_{\text{oos}}}{T+T_{\text{oos}}} \leqslant \frac{1}{4}-\frac{\epsilon}{2}.$$

By definition,

$$|D| = T_{\text{subs}} + T_{\text{oos}}.$$

On the other hand, the defining equation (35) of virtual length reveals that $\|Z\|$ for any set Z is the total number of transmissions in Z plus the total number of out-of-sync attacks in Z. In particular,

$$\|[1,T]\| = T + T_{\text{oos}}.$$

The last three equations immediately give (47). ■

G. Finish times

Let $e_1,e_2,\ldots,e_n$ be the edges of the unique root-to-leaf path in $X\cup Y$, listed in increasing order of depth. For $i=1,2,\ldots,n$, define f_i to be the index of the first transmission when e_i is sent (whether or not that transmission is subject to an attack). If e_i is never sent, we define $f_i=\infty$. For notational convenience, we also define $f_0=f_{-1}=f_{-2}=\cdots=0$. Recall from the description of the interactive coding scheme that Alice never sends an edge e unless she has previously sent all proper predecessors of e in X, and analogously for Bob. This gives

$$f_1 \leqslant f_3 \leqslant f_5 \leqslant \cdots,$$
$$f_2 \leqslant f_4 \leqslant f_6 \leqslant \cdots.$$

The overall sequence $f_1, f_2, f_3, f_4, f_5, f_6, \ldots$ need not be in sorted order, however, due to interference by the adversary. We abbreviate

$$\overline{f_i} = \max\{0, f_1, f_2, \ldots, f_i\}.$$

By basic arithmetic,

$$[\overline{f_{i-1}}, \overline{f_i}) = [\overline{f_{i-1}}, f_i), \qquad i = 1, 2, \ldots, n. \tag{48}$$

We now bound the virtual length of any such interval in terms of the number of bad transmissions in it, thereby showing that Alice and Bob make rapid progress as long as they do not experience too many attacks.

Lemma 29: For any integers i and t with $\overline{f_{i-1}} \leqslant t < f_i$,

$$|||[\overline{f_{i-1}}, t]||| \leqslant 2|B|_{[\overline{f_{i-1}}, t]} + 3. \tag{49}$$

Proof: We will only treat the case of i odd; the proof for even i can be obtained by swapping the roles of Alice and Bob below.

Consider any transmission $j \in E' \cap [\overline{f_{i-1}}, f_i)$. Lemma 21 ensures that on receipt of transmission j, Alice is able to correctly recover the set of edges that Bob has sent her by that time, which includes $e_2, e_4, e_6, \ldots, e_{i-1}$. At that same time, Alice has sent Bob $e_1, e_3, e_5, \ldots, e_{i-2}$ but not e_i, as one can verify from $j \in [\overline{f_{i-1}}, f_i)$. Therefore, the arrival of transmission j causes Alice either to exit or to immediately send e_i. Either way, the interval $[\overline{f_{i-1}}, f_i)$ does not contain any transmissions numbered $j+1$ or higher. We conclude that there is at most one transmission in $E' \cap [\overline{f_{i-1}}, f_i)$, and in particular

$$|E'|_{[\overline{f_{i-1}}, t]} \leqslant 1.$$

This upper bound directly implies (49) by Lemma 27. ■

H. The progress lemma

We have reached the technical centerpiece of our analysis. The result that we are about to prove shows that any sufficiently long execution of the interactive coding scheme with a sufficiently low virtual corruption rate allows Alice and Bob to exchange all the n edges of the unique root-to-leaf path in $X \cup Y$, and moreover this progress is not "undone" by any subsequent attacks by the adversary. The proof uses amortized analysis in an essential way.

Lemma 30 (Progress lemma): Let $t \in \{1, 2, \ldots, T\}$ be given with

$$|||[1, t]||| \geqslant \frac{n+2}{\alpha}, \tag{50}$$

$$\operatorname{corr}[1, t] \leqslant \frac{1}{4} - \alpha. \tag{51}$$

Then there is an integer $t^* \leqslant t$ such that

$$[\overline{f_n}, t^*) \cap E' \neq \varnothing, \tag{52}$$
$$[\overline{f_n}, t^*) \cap E'' \neq \varnothing, \tag{53}$$
$$\Delta(S'[i, t], R'[i, t]) < 1 - \alpha, \qquad i = 1, 2, \ldots, t^*, \tag{54}$$
$$\Delta(S''[i, t], R''[i, t]) < 1 - \alpha, \qquad i = 1, 2, \ldots, t^*. \tag{55}$$

Proof: Equations (54) and (55) hold vacuously for $t^* = 0$. In what follows, we will take $t^* \in \{0, 1, 2, \ldots, t\}$ to be the *largest* integer for which (54) and (55) hold. For the sake of contradiction, assume that at least one of the remaining desiderata (52), (53) is violated, whence

$$|||[\overline{f_n}, t^*)||| \leqslant 2|B|_{[\overline{f_n}, t^*)} + 1 \tag{56}$$

by Lemma 27. The proof strategy is to show that (56) is inconsistent with the hypothesis of the lemma. To this end, let $n^* \in \{0, 1, 2, \ldots, n\}$ be the largest integer such that $\overline{f_{n^*}} \leqslant t^*$. Then we have the partition

$$[0, t] = [\overline{f_0}, \overline{f_1}) \cup [\overline{f_1}, \overline{f_2}) \cup \cdots \\ \cup [\overline{f_{n^*-1}}, \overline{f_{n^*}}) \cup [\overline{f_{n^*}}, t^*) \cup \{t^*\} \cup (t^*, t].$$

The bulk of our proof is concerned with bounding the virtual length of each of the intervals on the right-hand side.

To begin with,

$$\begin{aligned} |||[\overline{f_{i-1}}, \overline{f_i})||| &= |||[\overline{f_{i-1}}, f_i)||| \\ &\leqslant 2|B|_{[\overline{f_{i-1}}, f_i)} + 3 \\ &\leqslant 2|B|_{[\overline{f_{i-1}}, \overline{f_i})} + 3 \end{aligned} \tag{57}$$

for any $i = 1, 2, \ldots, n^*$, where the first and third steps use (48), and the second step follows from Lemma 29. Next, the upper bound

$$|||[\overline{f_{n^*}}, t^*)||| \leqslant 2|B|_{[\overline{f_{n^*}}, t^*)} + 3 \tag{58}$$

follows from Lemma 29 if $n^* < n$ and from (56) if $n^* = n$. The virtual length of the singleton interval $\{t^*\}$ can be bounded from first principles:

$$|||\{t^*\}||| \leqslant 2. \tag{59}$$

Finally, recall from the definition of t^* that either $\max\{\Delta(S'[t^*+1, t], R'[t^*+1, t]), \Delta(S''[t^*+1, t], R''[t^*+1, t])\} \geqslant 1 - \alpha$ or $t^* = t$, leading to

$$|||(t^*, t]||| \leqslant \frac{4}{1-\alpha}|D|_{(t^*, t]} + 1 \tag{60}$$

by Lemma 26 in the former case and trivially in the latter.

Putting everything together, we obtain

$$\begin{aligned} &|||[1, t]||| \\ &\leqslant 2|B|_{[0, t^*)} + 3(n^*+1) + 2 + \frac{4}{1-\alpha}|D|_{(t^*, t]} + 1 \\ &\leqslant \frac{4}{1-\alpha}|D|_{[0, t^*)} + 3(n^*+1) + 2 + \frac{4}{1-\alpha}|D|_{(t^*, t]} + 1 \\ &\leqslant \frac{4}{1-\alpha}|D|_{[0, t]} + 3n + 6 \\ &\leqslant \frac{4}{1-\alpha}|D|_{[0, t]} + 3\alpha|||[1, t]|||, \end{aligned} \tag{61}$$

where the first step is the result of adding (57)–(60), the second step applies Lemma 23, and the final step uses (50). Since $0 < \alpha < 1$, the conclusion of (61) is equivalent to

$$\operatorname{corr}[1,t] \geqslant \frac{(1-3\alpha)(1-\alpha)}{4},$$

which is inconsistent with (51). We have obtained the desired contradiction and thereby proved that t^* satisfies each of the properties (52)–(55). ■

I. Finishing the proof

We have reached a "master theorem," which gives a sufficient condition for Alice and Bob to assign the correct value to their corresponding copies of the *out* variable. Once established, this result will allow us to easily finish the proof of Theorem 17.

Theorem 31: Consider a point in time when Alice updates her out variable, and fix a corresponding integer $t \leqslant T$ such that $\not*(R'[1,t])$ is the complete sequence of symbols that Alice has received by that time. Assume that

$$|\!|\!|[1,t]|\!|\!| \geqslant \frac{n+2}{\alpha}, \tag{62}$$

$$\operatorname{corr}[1,t] \leqslant \frac{1}{4} - \alpha. \tag{63}$$

Then as a result of the update, out is assigned the leaf vertex in the unique root-to-leaf path in $X \cup Y$. An analogous theorem holds for Bob.

Observe that Theorem 31 makes no assumption as to the actual timing of the update to *out*. It may happen that the update takes place in response to the t^{th} transmission; but it may also take place significantly earlier, due to out-of-sync attacks.

Proof of Theorem 31: We will only prove the claim for Alice; the proof of Bob is entirely analogous. Lemma 30 implies the existence of $j' \in E'$ and $j'' \in E''$ such that

$$\overline{f_n} \leqslant j' < t, \tag{64}$$

$$\overline{f_n} \leqslant j'' < t, \tag{65}$$

$$\Delta(S'[j'+1,t], R'[j'+1,t]) < 1-\alpha, \tag{66}$$

$$\Delta(S''[j''+1,t], R''[j''+1,t]) < 1-\alpha. \tag{67}$$

By the definition of E' and E'',

$$\operatorname{SD}(S'[1,j'], R'[1,j']) < 1-\alpha, \tag{68}$$

$$\operatorname{SD}(S''[1,j''], R''[1,j'']) < 1-\alpha. \tag{69}$$

As a result,

$$\begin{aligned}
&\operatorname{SD}_{\overline{*}(S'[1,j'])}(S'[1,t], R'[1,t]) \\
&\quad = \operatorname{SD}_{\overline{*}(S'[1,j'])}(S'[1,j']S'[j'+1,t],\ R'[1,j']R'[j'+1,t]) \\
&\quad \leqslant \max\{\operatorname{SD}_{\overline{*}(S'[1,j'])}(S'[1,j'], R'[1,j']), \\
&\qquad\qquad \Delta(S'[j'+1,t], R'[j'+1,t])\} \\
&\quad \leqslant \max\{\operatorname{SD}(S'[1,j'], R'[1,j']), \\
&\qquad\qquad \Delta(S'[j'+1,t], R'[j'+1,t])\} \\
&\quad < 1-\alpha,
\end{aligned} \tag{70}$$

where the second step is valid by Proposition 12, item 4); the third step uses (6); and the final step is immediate from (66) and (68).

When Alice updates her *out* variable, the sequence of symbols that she has received is $\not*(R'[1,t])$. By (70) and Theorem 15,

$$\begin{aligned}
\textsc{Decode}_{C,\alpha}(\not*(R'[1,t])) &\succeq (\not*(S'[1,t]))_{\leqslant\overline{*}(S'[1,j'])} \\
&= \not*(S'[1,j']).
\end{aligned}$$

Therefore, just prior to updating *out*, Alice is able to correctly recover the prefix $\not*(S'[1,j'])$ of the sequence of symbols sent to her by Bob. By Lemma 20, this means that she correctly recovers the complete set of edges encoded by the string $\not*(S'[1,j'])$. By (64), this prefix $\not*(S'[1,j'])$ contains the encoding of every edge of Y that appears in the root-to-leaf path in $X \cup Y$. Moreover, every edge encoded in $\not*(S'[1,j'])$ is correct in that it is an element of Y. Alice's pseudocode now ensures that she assigns to *out* the leaf vertex on the unique root-to-leaf path in $X \cup Y$.

The proof for Bob is entirely analogous, with (65), (67), (69), and j'' playing the role of (64), (66), (68), and j', respectively. ■

We are now in a position to establish the main result of this section.

Proof of Theorem 17: Recall that $n = |\pi|$ denotes the communication cost of the original protocol, and $\epsilon > 0$ is a constant in the statement of Theorem 17. Consider the interactive coding scheme given by Algorithms 2 and 3, with parameters set according to

$$\alpha = \frac{\epsilon}{4}, \tag{71}$$

$$N = \left\lceil \frac{n+4}{2\alpha} \right\rceil. \tag{72}$$

By (30), the coding scheme uses an alphabet of size at most $(|\Sigma| \cdot n/\epsilon)^{O(1/\epsilon)} = O(|\Sigma| \cdot n)^{O(1)} = O(|\Sigma| \cdot |\pi|)^{O(1)}$. Furthermore, by (31), the combined number of transmissions sent by Alice and Bob does not exceed $2N = O(n) = O(|\pi|)$.

It remains to show that when the communication stops, *out* is set for both Alice and Bob to the leaf vertex on the unique root-to-leaf path in $X \cup Y$. To this end, note from (71) and Lemma 28 that

$$\operatorname{corr}[1,T] \leqslant \frac{1}{4} - 2\alpha. \tag{73}$$

By (72) and Lemma 25,

$$|\!|\!|[1,T]|\!|\!| > \frac{n+4}{\alpha} \tag{74}$$

and therefore

$$|\!|\!|[1,T-1]|\!|\!| > \frac{n+2}{\alpha}. \tag{75}$$

Also,

$$\begin{aligned}
\operatorname{corr}[1,T-1] &\leqslant \frac{|\!|\!|[1,T]|\!|\!|}{|\!|\!|[1,T-1]|\!|\!|} \cdot \operatorname{corr}[1,T] \\
&\leqslant \left(1 + \frac{2}{|\!|\!|[1,T-1]|\!|\!|}\right) \cdot \operatorname{corr}[1,T] \\
&\leqslant \left(1 + \frac{2\alpha}{n+2}\right) \cdot \left(\frac{1}{4} - 2\alpha\right) \\
&\leqslant \frac{1}{4} - \alpha,
\end{aligned} \tag{76}$$

where the third step uses (73) and (75). Now, consider the last time that Alice and Bob update their copies of out. The complete sequence of symbols that Alice has received at the time of her last update is $\not{*}(R'[1, T-1])$ or $\not{*}(R'[1, T])$. Likewise, the complete sequence of symbols that Bob has received at the time of his last update is $\not{*}(R''[1, T-1])$ or $\not{*}(R''[1, T])$. By (73)–(76) and Theorem 31, both players set out to the leaf vertex in the unique root-to-leaf path in $X \cup Y$. This completes the proof of Theorem 17. ∎

V. A CODING SCHEME WITH A CONSTANT-SIZE ALPHABET

In this section, we will adapt the proof of Theorem 17 to use an alphabet of constant size. This modification will yield the main result of this paper (Theorems 1 and 2), which we restate here for the reader's convenience.

Theorem 32: Fix an arbitrary constant $\epsilon > 0$, *and let* π *be an arbitrary protocol with alphabet* Σ. *Then there exists an interactive coding scheme for* π *with alphabet size* $O(1)$ *and communication cost* $O(|\pi| \log |\Sigma|)$ *that tolerates*

1) *corruption rate* $\frac{1}{4} - \epsilon$;
2) *any normalized corruption rate* $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$ *such that* $\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} \leqslant \frac{1}{4} - \epsilon$.

In Section V-H, we further generalize Theorem 32, item 1) to the setting when Alice and Bob need to be ready with their answers by a certain round (based on each player's own counting) rather than when the communication stops. In that setting, too, our interactive coding scheme is optimal and matches the lower bound due to Braverman et al. [5]. At a high level, our proofs of Theorem 32 and its generalization are similar to the proof of Theorem 17 in the previous section, and we will be able to reuse most of the auxiliary machinery developed there. The principal point of departure is a new way of encoding and transferring edges, which in turn requires subtle modifications to the amortized analysis.

A. Edge representation and transfer

We may assume without loss of generality that π is in canonical form, which can be achieved for any protocol at the expense of doubling its communication cost. Canonical form allows us to identify Alice's input with a set X of odd-depth edges of the protocol tree for π, and Bob's input with a set Y of even-depth edges. Execution of π corresponds to following the unique root-to-leaf path in $X \cup Y$, whose length we denote by

$$n = |\pi|.$$

Recall that our previous interactive coding scheme in Section IV involved Alice and Bob sending each other edges from their respective input sets X and Y, with each transmission representing precisely one such edge. The new coding scheme also amounts to Alice and Bob exchanging edges from their respective input sets. This time, however, any given transmission will contain information about as many as Λ^2 edges, where $\Lambda = \Lambda(\epsilon) > 0$ is a constant to be chosen later. Moreover, to accommodate the size restriction on the alphabet, the encoding of any given edge will now be split across multiple transmissions. We say that a transmission *fulfills* an edge e if it carries the last bit of e's encoding.

Our approach to the transfer of edges is inspired by the interactive coding schemes with constant-size alphabets due to Braverman and Rao [6] and Braverman et al. [5]. We adapt their transfer in several ways to support our more general setting and to make the overall proof simpler. A detailed technical exposition follows.

Edge encoding: We will keep the policy that Alice does not start sending an edge e unless she has already fulfilled all predecessors of e in X, and likewise for Bob. This makes it possible for the sender to encode an edge e by referring to the previously transmitted grandparent of e. Specifically, an edge is now encoded as a triple (m, j, σ), where m is the number of transmissions sent by the sender since his or her most recent transmission that fulfilled the grandparent of e; the index $j \in \{1, 2, 3, \ldots, \Lambda^2\}$ identifies that grandparent among the up to Λ^2 edges featured in that transmission; and $\sigma \in \Sigma \times \Sigma$ identifies e relative to that grandparent. As a base case, an edge of depth 1 or 2 is encoded by a triple (m, j, σ) where m is the number of transmissions sent by the sender since the beginning of time, and j is ignored. Note that how an edge is encoded is highly context-sensitive in that it depends on previous transmissions by the sender. As a result, whenever we speak of *the* encoding of an edge e, we are referring to the encoding of e at a particular time that will be clear from the context.

Ignoring inessential details, the key to this encoding scheme is that an edge e is specified by indicating how long ago its grandparent was fulfilled. This idea, introduced in [6] and used more recently in [5], helps keep encodings short on average.

Chunking: A constant-size alphabet makes it in general impossible to deliver the entire encoding of an edge in a single transmission. Instead, we split the encoding of every edge into *chunks.* A chunk contains a single bit of the encoding of the edge as well as 3 bits of metadata. Thus, the number of chunks needed to transfer an edge is equal to the bit length of e's encoding. Alice and Bob each maintain data structures called *encoding* and *numBitsSent*, indexed by edges. These data structures store, for each of the edges currently being transferred, its encoding and the number of bits sent so far.

Parallelism: Rather than send edges one by one, each player will send up to Λ^2 edges in parallel. To see the intuitive reason for doing so, consider the transfer of a typical edge e, which spans multiple transmissions. As Alice sends e chunk by chunk to Bob, she simultaneously receives information from him, which in turn may lead her to believe that she should be sending an edge other than e. The problem is, she can never be sure! Simply aborting the transfer of e is wasteful if e later turns out to be the right edge to send. Instead, we allow transfer of several edges in parallel and use an additional, credit-based mechanism for identifying and aborting unpromising transfers.

Specifically, each player maintains an ordered list L of edges that he or she is currently transferring. New edges are inserted in L at the front rather than back, reflecting that view that new information should be prioritized over old. To prepare a transmission, a player looks at the first Λ^2 edges in L and takes a chunk of each. If L has fewer than Λ^2 edges, the player simply takes a chunk of each edge in L. The concatenation of these chunks, ordered the same way as the corresponding edges in L, forms a *page*, which we view as a symbol from an auxiliary alphabet Σ_{in}. Since an edge chunk is a 4 bits long, the size of Σ_{in} is bounded by a constant:

$$|\Sigma_{\text{in}}| = \sum_{i=0}^{\Lambda^2} 2^{4i} = \frac{16^{\Lambda^2+1}-1}{15}. \tag{77}$$

Credit: As a crucial component of the transfer scheme, Alice and Bob each maintain a data structure called *credit*. This data structure is indexed by edges and stores the amount of "funds" available to pay for the transfer of any given edge e. The credit of every edge is initialized to 0 at the beginning, and remains nonnegative from then on. Every receive-send cycle identifies an edge e to send, which then gets a credit increase of Λ and is additionally inserted in L unless it is already there. Any time an edge chunk is sent, the credit of the corresponding edge is decreased by 1. An edge remains in L until its credit reaches 0 or until its last chunk is sent, whichever comes first. At that point, the edge is removed from L.

Metadata: The purpose of the metadata in each edge chunk is to allow the receiver to correctly piece together the encodings of the edges. A chunk for an edge e is always prepared at send time rather than in advance and includes the following four bits: the next bit of the encoding of e; a bit to indicate if this is the first chunk for e; a bit to indicate if this is the last chunk for e; and a bit to indicate if e's credit has reached zero. The last two bits alert the receiver to the removal of e from the sender's edge list.

B. The simulation

Algorithm 4 gives the pseudocode to support our edge encoding and transfer scheme. The pseudocode is identical for Alice and Bob. In the pseudocode, $\circ$ denotes string concatenation, $|L|$ denotes the number of edges in L, and $L[i]$ denotes the i^{th} edge in L. The edge operations are as follows.

1) ADDEDGE is executed once by each player during his or her receive-send cycle. As an argument, it receives an edge which that player wants to send next. If e is already on the player's edge list, ADDEDGE simply increments e's credit by Λ. If not, ADDEDGE increments e's credit by Λ, computes an encoding of e relative to the player's current transmission count, and adds e to the edge list ahead of any existing edges.
2) NEXTCHUNK receives as an argument an edge e and returns the next 4-bit chunk of that edge, based on the stored encoding of e and the number of bits of e's encoding sent so far. This procedure uses *numBitsSent*(e), *credit*(e), and *encoding*(e) to correctly set the metadata for the chunk. It then updates *numBitsSent*(e) and *credit*(e) to reflect the remaining number of bits to send and the edge's available credit.
3) NEXTPAGE is the procedure that assembles the next page to send. The page is made up of at most Λ^2 chunks, one for each of the first Λ^2 edges on the edge list. The chunks are prepared using NEXTCHUNK. Once the page is assembled, NEXTPAGE updates the edge list by removing edges that have been fully sent or have no credit left.

Algorithm 4: Edge operations

```
Global variables: encoding, numBitsSent, credit, L
Procedure AddEdge($e, i$)
    $credit(e) \leftarrow credit(e) + \Lambda$
    if $e \notin L$ then
        $encoding(e) \leftarrow$ encoding of $e$ based on current
          transmission count $i$
        $numBitsSent(e) \leftarrow 0$
        prepend $e$ to $L$, ahead of any existing edges
    end
Procedure NextChunk($e$)
                    // Update edge statistics
    $numBitsSent(e) \leftarrow numBitsSent(e) + 1$
    $credit(e) \leftarrow credit(e) - 1$
                    // Compute edge chunk
    return $(encoding(e))_{numBitsSent(e)}$
            $\circ\ \mathbf{I}[numBitsSent(e) = 1]$
            $\circ\ \mathbf{I}[numBitsSent(e) = |encoding(e)|]$
            $\circ\ \mathbf{I}[credit(e) = 0]$
Procedure NextPage()
    $page \leftarrow$ NextChunk($L[1]$)
            $\circ$ NextChunk($L[2]$)
            $\circ \cdots$
            $\circ$ NextChunk($L[\min\{\Lambda^2, |L|\}]$)
                    // Clean up the edge list
    foreach $e \in L$ do
        if $credit(e) = 0$ or
          $numBitsSent(e) = |encoding(e)|$ then
            remove $e$ from $L$
        end
    end
    return $page$
```

The overall interactive coding scheme is given by Algorithms 5 and 6 for Alice and Bob, respectively. The main novelty relative to the scheme of Section IV are the calls to ADDEDGE and NEXTPAGE, which a player executes as soon as he or she has identified an edge e to send. Apart from that, the remarks made in Section IV apply here in full. In particular, $\alpha = \alpha(\epsilon) \in (0,1)$ and $N = N(n,\alpha)$ are parameters to be chosen later. We set

$$\Lambda = \left\lceil \frac{2}{\alpha} \right\rceil \tag{78}$$

and fix an arbitrary α-good code $C\colon \Sigma_{\text{in}}^* \to \Sigma_{\text{out}}^*$ whose existence is ensured by Theorem 9. That theorem implies, in view of (77) and (78), that

$$|\Sigma_{\text{out}}| \leqslant 2^{O(1/\alpha^3)}. \tag{79}$$

Alice and Bob use C to encode every transmission. In particular, the encoded symbol from Σ_{out} at any given point depends

Algorithm 5: Coding scheme for Alice

```
Input: $X$ (set of Alice's edges)
1  $L \leftarrow \varnothing$
2  $credit(e) \leftarrow 0$ for every edge $e$
3  $e \leftarrow$ the edge in $X$ incident to the root
4  AddEdge$(e, 1)$
5  $page \leftarrow$ NextPage()
6  encode and send $page$
7  foreach $i = 1, 2, 3, \dots, N$ do
8      receive a symbol $r_i \in \Sigma_{\text{out}}$
9      $s \leftarrow$ Decode$_{C,\alpha}(r_1 r_2 \dots r_i)$
10     interpret $s$ as a sequence $B$ of even-depth edges
11     $\ell \leftarrow$ maximum length of a rooted path in $X \cup B$
12     compute the shortest prefix of $B$ s.t. $X \cup B$ contains
         a rooted path of length $\ell$, and let $P$ be the rooted
         path so obtained
13     $out \leftarrow$ deepest vertex in $P$
14     if $i \leqslant N - 1$ then
15         $e \leftarrow$ the deepest edge in $P \cap X$ whose proper
             predecessors in $X$ have all been sent
16         AddEdge$(e, i + 1)$
17         $page \leftarrow$ NextPage()
18         encode and send $page$
19     end
20 end
```

Algorithm 6: Coding scheme for Bob

```
Input: $Y$ (set of Bob's edges)
1  $L \leftarrow \varnothing$
2  $credit(e) \leftarrow 0$ for every edge $e$
3  foreach $i = 1, 2, 3, \dots, N$ do
4      receive a symbol $r_i \in \Sigma_{\text{out}}$
5      $s \leftarrow$ Decode$_{C,\alpha}(r_1 r_2 \dots r_i)$
6      interpret $s$ as a sequence $A$ of odd-depth edges
7      $\ell \leftarrow$ maximum length of a rooted path in $Y \cup A$
8      compute the shortest prefix of $A$ s.t. $Y \cup A$ contains
         a rooted path of length $\ell$, and let $P$ be the rooted
         path so obtained
9      $out \leftarrow$ deepest vertex in $P$
10     $e \leftarrow$ the deepest edge in $P \cap Y$ whose proper
         predecessors in $Y$ have all been sent
11     AddEdge$(e, i)$
12     $page \leftarrow$ NextPage()
13     encode and send $page$
14 end
```

not only on the symbol from Σ_{in} being transmitted but also on the content of the previous transmissions by the sender. The decoding is again done using the Decode$_{C,\alpha}$ algorithm of Theorem 15. Note from the pseudocode that Alice and Bob send at most N transmissions each.

It remains to elaborate on the decoding and interpretation steps in the interactive coding scheme. To do so, we first prove that the sequence of pages sent by one of the players at any given point reveals the sequence of edges that that player has fulfilled so far.

Lemma 33: Consider an arbitrary point in time, and let $p_1, p_2, \dots, p_t \in \Sigma_{\text{in}}$ be the sequence of pages sent by one of the players so far. Then that sequence uniquely identifies the corresponding sequence of edges $e_1, e_2, \dots, e_{t'}$ fulfilled by that player.

Proof: We first reconstruct as completely as possible the sender's state at the times when each of the pages $p_1, p_2, \dots, p_t$ has just been assembled. Specifically, we determine the length of the sender's edge list, the transmission status of every edge on the edge list (in progress, aborted, or fulfilled), and the corresponding part of the encoding transferred for every edge so far. This reconstruction process involves working inductively through the page sequence $p_1, p_2, \dots, p_t$ and using the metadata to identify when an edge is new, in progress, aborted, or fulfilled. Recall that there is at most one new edge per page, and it is always inserted at the *front* of the edge list.

The first stage reconstructs the complete list of edge encodings sent so far by the sender, along with the final status of each encoding (in progress, aborted, or fulfilled), and the start and end times of each fulfilled encoding. We then interpret the fulfilled encodings as a sequence $(m_1, j_1, \sigma_1), (m_2, j_2, \sigma_2), \dots, (m_{t'}, j_{t'}, \sigma_{t'})$ of edge representations. Using the end times of the fulfilled encodings and their indices inside the pages than fulfilled them, we can reconstruct the corresponding sequence of edges $e_1, e_2, \dots, e_{t'}$ via an inductive process analogous to that in Lemma 18. ■

With Lemma 33 in hand, the decoding and interpretation steps in lines 9–10 for Alice and lines 5–6 for Bob are implemented the same way they were for a large alphabet. Specifically, the decoding step produces a codeword s of C, which by Fact 8 corresponds to a unique string in Σ_{in}^*. This string is by definition a sequence of pages $p_1, p_2, p_3, \dots$, from which the receiving party can reconstruct the corresponding sequence of fulfilled edges using the inductive procedure of Lemma 33. It may happen that the page sequence $p_1, p_2, p_3, \dots$ is syntactically malformed; in that case, the receiving party interrupts the interpretation process at the longest prefix of $p_1, p_2, p_3, \dots$ that corresponds to a legitimate sequence of edges. This completes the interpretation step, yielding a sequence of edges A for Bob and B for Alice.

Analogous to the interactive coding scheme of Section IV, Alice and Bob each maintain a variable called *out*. In Sections V-C–V-G below, we will examine an arbitrary but fixed execution of the interactive coding scheme. In particular, we will henceforth consider the inputs X and Y and the adversary's actions to be fixed. We allow any behavior by the adversary as long as it meets one of the criteria 1), 2) in Theorem 32. We will show that as soon as the communication stops, *out* is set for both Alice and Bob to the leaf vertex of the unique root-to-leaf path in $X \cup Y$. This will prove Theorem 32.

C. Fundamental notions and facts

We adopt the notation and definitions of Sections IV-B–IV-F in their entirety. These items carry over without any changes because they pertain to the lowest level of abstraction (the "data link layer," as it were), which cannot distinguish between the old and new interactive coding schemes. As a consequence, all results proved in Sections IV-B–IV-F apply here in full, with the exception are Lemmas 20 and 21 whose wording needs to be clarified by replacing "sent edges" with "fulfilled edges." The result of this cosmetic modification is as follows.

Lemma 34: Let $t \in \{1, 2, \ldots, T\}$ be given. Then:

1) *the string $\not\ast(S'[1, t])$ uniquely identifies the sequence of protocol tree edges that Bob fulfills over the course of transmissions $1, 2, \ldots, t$;*
2) *the string $\not\ast(S''[1, t])$ uniquely identifies the sequence of protocol tree edges that Alice fulfills over the course of transmissions $1, 2, \ldots, t$.*

Proof: By symmetry, it suffices to prove the former claim. By Fact 8, the codeword $\not\ast(S'[1, t]) \in \Sigma_{\text{out}}^*$ corresponds to a unique string in Σ_{in}^*, which is the sequence of pages that Bob sends Alice over the course of the first t transmissions. By Lemma 33, this sequence of pages uniquely identifies the corresponding fulfilled edges. ■

Lemma 35: Let $t \in \{1, 2, \ldots, T\}$ be given.

1) *If $t \in E'$, then on receipt of transmission t, Alice is able to correctly recover the complete sequence of edges that Bob has fulfilled by that time.*
2) *If $t \in E''$, then on receipt of transmission t, Bob is able to correctly recover the complete sequence of edges that Alice has fulfilled by that time.*

Proof: By symmetry, it again suffices to prove the former claim. Let $t \in E'$. Then by definition, $\text{SD}(S'[1, t], R'[1, t]) < 1 - \alpha$. Taking $k = \infty$ in Theorem 15, we conclude that $\text{DECODE}_{C,\alpha}(\not\ast(R'[1, t])) = \not\ast(S'[1, t])$. This means that on receipt of transmission t, Alice is able to correctly recover the entire codeword $\not\ast(S'[1, t])$ that Bob has sent her so far. By Lemma 34, this in turn makes it possible for Alice to correctly identify the corresponding sequence of fulfilled edges. ■

D. Full pages

Recall that a page can contain at most Λ^2 edge chunks. If a page contains exactly Λ^2 chunks, we call it *full*. We define $F' \subseteq \{1, 2, \ldots, T\}$ as the set of transmissions where Alice sends a full page, and analogously $F'' \subseteq \{1, 2, \ldots, T\}$ as the set of transmissions where Bob sends a full page. In other words,

$$F' = \{i : S''[i, i] \text{ is a full page}\},$$
$$F'' = \{i : S'[i, i] \text{ is a full page}\}.$$

We abbreviate

$$F = F' \cup F''.$$

The following lemma, due to Braverman et al. [5, Lemma D.1], shows that full pages are relatively uncommon.

Proposition 36 (Braverman et al.): For any interval J such that $1 \in J$,

$$|F|_J \leqslant \frac{|J \cap \{1, 2, 3, \ldots, T\}|}{\Lambda}.$$

Proof (adapted from Braverman et al.): Since $F \subseteq \{1, 2, \ldots, T\}$, the proposition is equivalent to the following statement:

$$|F \cap \{1, 2, \ldots, t\}| \leqslant \frac{t}{\Lambda}$$

for all $1 \leqslant t \leqslant T$. The proof proceeds by a potential argument. The potential function to consider is the sum of the credit values of Alice's edges. This quantity is always nonnegative and is initially zero. Any full page sent by Alice causes a decrement of the credit counter for each edge in the page, decreasing the potential function by Λ^2. On the other hand, any increase in the potential function is due to the arrival of a symbol (i.e., a good event or insertion addressed to Alice) and is precisely Λ. Since the potential function is nonnegative, we conclude that

$$\Lambda |F' \cap \{1, 2, \ldots, t\}| \leqslant |(G' \cup I') \cap \{1, 2, \ldots, t\}|.$$

Analogously,

$$\Lambda |F'' \cap \{1, 2, \ldots, t\}| \leqslant |(G'' \cup I'') \cap \{1, 2, \ldots, t\}|.$$

Therefore,

$$\begin{aligned} \Lambda |F \cap \{1,2, \ldots, t\}| & \\ &\leqslant \Lambda |F' \cap \{1, 2, \ldots, t\}| + \Lambda |F'' \cap \{1, 2, \ldots, t\}| \\ &\leqslant |(G' \cup I') \cap \{1, 2, \ldots, t\}| \\ &\qquad + |(G'' \cup I'') \cap \{1, 2, \ldots, t\}|. \end{aligned} \tag{80}$$

Items 1), 2), and 7) of Lemma 19 imply that G', G'', I', I'' are pairwise disjoint. Therefore, the sum on the right-hand side of (80) does not exceed t. ■

E. Finish times

We adopt the notation and definitions of Section IV-G, and review them here for the reader's convenience. Let $e_1, e_2, \ldots, e_n$ be the edges of the unique root-to-leaf path in $X \cup Y$, listed in increasing order of depth. Recall that a transmission *fulfills* an edge e if the corresponding page sent by the sender carries the last bit of an encoding of e. For $i = 1, 2, \ldots, n$, define f_i to be the index of the first transmission that fulfills e_i (whether or not that transmission is subject to an attack). If e_i is never fulfilled, we set $f_i = \infty$. For notational convenience, we also define $f_0 = f_{-1} = f_{-2} = \cdots = 0$. Recall from the description of the interactive coding scheme that Alice never starts sending an edge e unless she has finished sending all proper predecessors of e in X, and analogously for Bob. This gives

$$f_1 \leqslant f_3 \leqslant f_5 \leqslant \cdots,$$
$$f_2 \leqslant f_4 \leqslant f_6 \leqslant \cdots.$$

The overall sequence $f_1, f_2, f_3, f_4, f_5, f_6, \ldots$ need not be in sorted order, however, due to interference by the adversary. We abbreviate

$$\overline{f_i} = \max\{0, f_1, f_2, \ldots, f_i\}.$$

By basic arithmetic,

$$[\overline{f_{i-1}}, \overline{f_i}) = [\overline{f_{i-1}}, f_i), \qquad i = 1, 2, \ldots, n. \tag{81}$$

Analogous to the analysis in Section IV-G, we need to bound the virtual length of each interval $[\overline{f_{i-1}}, f_i)$. To this end, we first bound the bit length of any encoding of e_i.

Lemma 37: For given integers i and t, suppose that an encoding of e_i is computed prior to the sending of transmission t. Then that encoding has bit length at most

$$\lceil \log(t - f_{i-2}) \rceil + \lceil 2 \log \Lambda |\Sigma| \rceil.$$

Proof: Recall that e_i is encoded as a triple (m, j, σ), where m is the number of transmissions sent by the sender since his or her page that contained the last bit of e_{i-2} (for $i \geqslant 3$) or since the beginning of time (for $i = 1, 2$); $j \in \{1, 2, \ldots, \Lambda^2\}$ identifies e_{i-2} inside that page; and $\sigma \in \Sigma \times \Sigma$ identifies e_i relative to e_{i-2}. The pair (j, σ) takes on $\Lambda^2|\Sigma|^2$ distinct values and can therefore be represented by a binary string of fixed length $\lceil 2 \log \Lambda |\Sigma| \rceil$. The remaining component m is a nonnegative integer of magnitude at most $t - f_{i-2} - 1$ and can therefore be represented by a binary string of length $\lceil \log(t - f_{i-2}) \rceil$ in the usual manner: $\varepsilon, 1, 10, 11, 100, \ldots$ for $0, 1, 2, 3, 4, \ldots$, respectively. ∎

We are now in a position to analyze the virtual length of any interval $[\overline{f_{i-1}}, f_i)$. The lemma that we are about to prove is a counterpart of Lemma 29.

Lemma 38: For any $t \in \{1, 2, \ldots, T\}$ and i with $\overline{f_{i-1}} \leqslant t < f_i$,

$$\begin{aligned}\|\!|[\overline{f_{i-1}}, t]|\!\| \leqslant{}& \frac{2\Lambda}{\Lambda - 1}|B|_{[\overline{f_{i-1}}, t]} + 2|F|_{[\overline{f_{i-1}}, t]} \\ &+ 2\lceil \log(t - f_{i-2}) \rceil + 2\lceil 2 \log 2\Lambda |\Sigma| \rceil.\end{aligned}$$

Proof: We will only treat the case of i odd; the proof for even i can be obtained by swapping the roles of Alice and Bob below.

For an edge e of the protocol tree, let *credit*(e, j) denote the value of *credit*(e) on Alice's side at the moment when transmission j enters the communication channel, i.e., immediately after the sender of transmission j has executed NEXTPAGE. For notational convenience, we also define *credit*$(e, 0) = 0$ for all e. Let $s \in [\overline{f_{i-1}}, t + 1]$ be the smallest integer such that *credit*$(e_i, j) > 0$ for $j = s, s + 1, \ldots, t$. Then

$$[\overline{f_{i-1}}, t] \subseteq [\overline{f_{i-1}}, s - 1) \cup [s - 1, s) \cup [s, t].$$

With this in mind, we complete the proof of the lemma by bounding the virtual length of each interval on the right-hand side and summing the resulting bounds. Key to our analysis are the following two claims.

Claim 39: $|E'|_{[\overline{f_{i-1}}, s-1)} \leqslant |B'|_{[\overline{f_{i-1}}, s-1)} / (\Lambda - 1)$.

Proof: Consider any transmission $j \in E' \cap [\overline{f_{i-1}}, t)$. Lemma 35 ensures that on receipt of transmission j, Alice is able to correctly recover the complete set of edges that Bob has finished sending her by that time, which includes $e_2, e_4, e_6, \ldots, e_{i-1}$. At that same time, Alice has finished sending Bob $e_1, e_3, e_5, \ldots, e_{i-2}$ but not e_i, as one can verify from $\overline{f_{i-1}} \leqslant j < t < f_i$. Therefore, the arrival of transmission j causes Alice to increase the credit of e_i by Λ in the call to ADDEDGE. The subsequent call to NEXTPAGE either leaves e_i's credit unchanged or decreases it by 1.

We now return to the proof of the claim. If $[\overline{f_{i-1}}, s - 1) = \varnothing$, the claim holds trivially. In the complementary case, the definition of s ensures that

$$\textit{credit}(e_i, s - 1) = 0. \tag{82}$$

By the previous paragraph, the net effect of an incoming excellent transmission in the interval $[\overline{f_{i-1}}, t)$ is to increase e_i's credit by at least $\Lambda - 1$, whereas none of the other incoming symbols decrease e_i's credit by more than 1. Since credit is always nonnegative, we conclude from (82) that the number of incoming excellent transmissions in the interval $[\overline{f_{i-1}}, s - 1)$ is at most a $1/(\Lambda - 1)$ fraction of the number of Alice's other incoming symbols in that interval. Formulaically, this conclusion translates to

$$|E'|_{[\overline{f_{i-1}}, s-1)} \leqslant \frac{1}{\Lambda - 1}|(G' \cup I') \setminus E'|_{[\overline{f_{i-1}}, s-1)},$$

which is equivalent to the claimed inequality by the definition of B'. ∎

Claim 40: $|G'' \cup D''|_{[s,t]} \leqslant |F'|_{[s,t]} + \lceil \log(t - f_{i-2}) \rceil + \lceil 2 \log \Lambda |\Sigma| \rceil$.

Proof: By the choice of s, the credit of e_i is positive when transmissions $s, s + 1, \ldots, t$ enter the communication channel. Since Alice does not fulfill e_i before or during transmission $t < f_i$, we conclude that e_i is *continuously* present on Alice's edge list as transmissions $s, s + 1, \ldots, t$ are prepared by their respective senders. In particular, every transmission among $s, s + 1, \ldots, t$ that is sent by Alice must contain a bit of the encoding of e_i unless it is a full page. We conclude that

$$|G'' \cup D''|_{[s,t]} \leqslant |F'|_{[s,t]} + L,$$

where by definition $G'' \cup D''$ is the set of transmissions sent by Alice, F' is the set of transmissions sent by Alice that are full pages, and L stands for the bit length of e_i's encoding. This completes the proof in view of the upper bound on L in Lemma 37. ∎

It remains to put everything together. We have

$$\begin{aligned}&\|\!|[\overline{f_{i-1}}, s - 1)|\!\| \\ &\quad \leqslant 2(|B|_{[\overline{f_{i-1}}, s-1)} + |E'|_{[\overline{f_{i-1}}, s-1)}) + 1 \\ &\quad \leqslant 2\left(|B|_{[\overline{f_{i-1}}, s-1)} + \frac{1}{\Lambda - 1}|B'|_{[\overline{f_{i-1}}, s-1)}\right) + 1 \\ &\quad \leqslant \frac{2\Lambda}{\Lambda - 1}|B|_{[\overline{f_{i-1}}, s-1)} + 1,\end{aligned} \tag{83}$$

where the first and second steps follow from Lemma 27 and Claim 39, respectively. Similarly,

$$\begin{aligned}\|\!|[s,t]|\!\| &\leqslant 2(|B|_{[s,t]}+|E''|_{[s,t]})+1\\ &\leqslant 2(|B|_{[s,t]}+|G''|_{[s,t]})+1\\ &\leqslant 2(|B|_{[s,t]}+|F'|_{[s,t]}+\lceil\log(t-f_{i-2})\rceil\\ &\qquad+\lceil 2\log\Lambda|\Sigma|\rceil)+1\\ &\leqslant 2(|B|_{[s,t]}+|F|_{[\overline{f_{i-1}},t]}+\lceil\log(t-f_{i-2})\rceil\\ &\qquad+\lceil 2\log\Lambda|\Sigma|\rceil)+1, \qquad (84)\end{aligned}$$

where the first and second steps follow from Lemma 27 and Claim 40, respectively. Finally,

$$\begin{aligned}\|\!|[s-1,s)|\!\| &= \|\!|\{s-1\}|\!\|\\ &\leqslant 2. \qquad (85)\end{aligned}$$

Adding the bounds in (83)–(85) proves the lemma. ■

F. The progress lemma

We have reached the technical centerpiece of our analysis, which is the counterpart of Lemma 30 for a large alphabet. Analogous to that earlier lemma, the result that we are about to prove shows that any sufficiently long execution of the interactive coding scheme with a sufficiently low virtual corruption rate allows Alice and Bob to exchange all the n edges of the unique root-to-leaf path in $X\cup Y$, and moreover this progress is not "undone" by any subsequent attacks by the adversary. Our exposition below emphasizes the similarities between Lemma 30 and the new result.

Lemma 41 (Progress lemma): Let $t\in\{1,2,\ldots,T\}$ *be given with*

$$\|\!|[1,t]|\!\| \geqslant \frac{cn}{\alpha}\log\frac{|\Sigma|}{\alpha}, \qquad (86)$$

$$\operatorname{corr}[1,t]\leqslant\frac{1}{4}-\alpha, \qquad (87)$$

where $c\geqslant 1$ *is a sufficiently large absolute constant. Then there is an integer* $t^*\leqslant t$ *such that*

$$[\overline{f_n},t^*)\cap E'\neq\varnothing, \qquad (88)$$

$$[\overline{f_n},t^*)\cap E''\neq\varnothing, \qquad (89)$$

$$\Delta(S'[i,t],R'[i,t])<1-\alpha, \qquad i=1,2,\ldots,t^*, \qquad (90)$$

$$\Delta(S''[i,t],R''[i,t])<1-\alpha, \qquad i=1,2,\ldots,t^*. \qquad (91)$$

Proof: Equations (90) and (91) hold vacuously for $t^*=0$. In what follows, we will take $t^*\in\{0,1,2,\ldots,t\}$ to be the *largest* integer for which (90) and (91) hold. For the sake of contradiction, assume that at least one of the remaining desiderata (88), (89) is violated, whence

$$\|\!|[\overline{f_n},t^*)|\!\|\leqslant 2|B|_{[\overline{f_n},t^*)}+1 \qquad (92)$$

by Lemma 27. The proof strategy is to show that (92) is inconsistent with the hypothesis of the lemma. To this end, let $n^*\in\{0,1,2,\ldots,n\}$ be the largest integer such that $\overline{f_{n^*}}\leqslant t^*$. Then we have the partition

$$\begin{aligned}[0,t] = [\overline{f_0},\overline{f_1})\cup[\overline{f_1},\overline{f_2})\cup\cdots\\ \cup[\overline{f_{n^*-1}},\overline{f_{n^*}})\cup[\overline{f_{n^*}},t^*)\cup\{t^*\}\cup(t^*,t].\end{aligned}$$

The bulk of our proof is concerned with bounding the virtual length of each of the intervals on the right-hand side.

Abbreviate

$$M=2\lceil 2\log 2\Lambda|\Sigma|\rceil+2. \qquad (93)$$

Then

$$\begin{aligned}\|\!|[\overline{f_{i-1}},\overline{f_i})|\!\| &= \|\!|[\overline{f_{i-1}},f_i)|\!\|\\ &\leqslant\frac{2\Lambda}{\Lambda-1}|B|_{[\overline{f_{i-1}},f_i)}+2|F|_{[\overline{f_{i-1}},f_i)}\\ &\qquad+2\log(f_i-f_{i-2})+M\\ &\leqslant\frac{2\Lambda}{\Lambda-1}|B|_{[\overline{f_{i-1}},\overline{f_i})}+2|F|_{[\overline{f_{i-1}},\overline{f_i})}\\ &\qquad+2\log(f_i-f_{i-2})+M \qquad (94)\end{aligned}$$

for any $i=1,2,\ldots,n^*$, where the first and third steps use (81), and the second step follows from Lemma 38. Next, the upper bound

$$\begin{aligned}\|\!|[\overline{f_{n^*}},t^*)|\!\| &\leqslant\frac{2\Lambda}{\Lambda-1}|B|_{[\overline{f_{n^*}},t^*)}+2|F|_{[\overline{f_{n^*}},t^*)}\\ &\qquad+2\log(t^*-f_{n^*-1})+M \qquad (95)\end{aligned}$$

follows from Lemma 38 if $n^*<n$ and from (92) if $n^*=n$. The virtual length of the singleton interval $\{t^*\}$ can be bounded from first principles:

$$\|\!|\{t^*\}|\!\|\leqslant 2. \qquad (96)$$

Finally, recall from the definition of t^* that either $\max\{\Delta(S'[t^*+1,t],R'[t^*+1,t]),\ \Delta(S''[t^*+1,t],R''[t^*+1,t])\}\geqslant 1-\alpha$ or $t^*=t$, leading to

$$\|\!|(t^*,t]|\!\|\leqslant\frac{4}{1-\alpha}|D|_{(t^*,t]}+1 \qquad (97)$$

by Lemma 26 in the former case and trivially in the latter.

It remains to put together the upper bounds in (94)–(97).

We have

$$\begin{aligned}
&|\!|\!|[1,t^*)|\!|\!| \qquad (98)\\
&\leqslant \frac{2\Lambda}{\Lambda-1}|B|_{[0,t^*)} + 2|F|_{[0,t^*)} + (n^*+1)M\\
&\qquad + 2\sum_{i=1}^{n^*}\log(f_i - f_{i-2}) + 2\log(t^* - f_{n^*-1})\\
&\leqslant \frac{2\Lambda}{\Lambda-1}\cdot\frac{2}{1-\alpha}|D|_{[0,t^*)} + \frac{2t^*}{\Lambda} + (n^*+1)M\\
&\qquad + 2\sum_{i=1}^{n^*}\log(f_i - f_{i-2}) + 2\log(t^* - f_{n^*-1})\\
&\leqslant \frac{2\Lambda}{\Lambda-1}\cdot\frac{2}{1-\alpha}|D|_{[0,t^*)} + \frac{2t^*}{\Lambda} + (n^*+1)M\\
&\qquad + 2(n^*+1)\log\frac{\sum_{i=1}^{n^*}(f_i - f_{i-2}) + (t^* - f_{n^*-1})}{n^*+1}\\
&= \frac{2\Lambda}{\Lambda-1}\cdot\frac{2}{1-\alpha}|D|_{[0,t^*)} + \frac{2t^*}{\Lambda} + (n^*+1)M\\
&\qquad + 2(n^*+1)\log\frac{f_{n^*} + t^*}{n^*+1}\\
&\leqslant \frac{2\Lambda}{\Lambda-1}\cdot\frac{2}{1-\alpha}|D|_{[0,t^*)} + \frac{2t^*}{\Lambda} + (n^*+1)M\\
&\qquad + 2(n^*+1)\log\frac{2t^*}{n^*+1}\\
&\leqslant \frac{4}{(1-\alpha)^2}|D|_{[0,t^*)} + \alpha|\!|\!|[1,t]|\!|\!|\\
&\qquad + 2(n^*+1)\cdot\left\lceil 1 + 2\log 2|\Sigma|\left\lceil\frac{2}{\alpha}\right\rceil\right\rceil\\
&\qquad + 2(n^*+1)\log\frac{2|\!|\!|[1,t]|\!|\!|}{n^*+1}\\
&\leqslant \frac{4}{(1-\alpha)^2}|D|_{[0,t^*)} + 2\alpha|\!|\!|[1,t]|\!|\!| - 3, \qquad (99)
\end{aligned}$$

where the first step follows from (94) and (95); the second step applies Lemmas 23 and 36; the third step is valid by the concavity of the logarithm function; the next-to-last step is immediate from (78), (93), and $t^* \leqslant t \leqslant |\!|\!|[1,t]|\!|\!|$; and the last step follows from (86) and $n^* \leqslant n$. Adding (96)–(99), we obtain

$$|\!|\!|[1,t]|\!|\!| \leqslant \frac{4}{(1-\alpha)^2}|D|_{[0,t]} + 2\alpha|\!|\!|[1,t]|\!|\!|,$$

or equivalently

$$\operatorname{corr}[1,t] \geqslant \frac{(1-2\alpha)(1-\alpha)^2}{4}.$$

This conclusion is inconsistent with (87) since $0 < \alpha < 1$. We have reached the desired contradiction and thereby proved that t^* satisfies each of the properties (88)–(91). ∎

G. Finishing the proof

We have reached a "master theorem" analogous to Theorem 31 for a large alphabet, which gives a sufficient condition for Alice and Bob to assign the correct value to their corresponding copies of the *out* variable. Once established, this new result will allow us to easily finish the proof of Theorem 32.

Theorem 42: Consider a point in time when Alice updates her out variable, and fix a corresponding integer $t \leqslant T$ such that $\not\!\!\;(R'[1,t])$ is the complete sequence of symbols that Alice has received by that time. Assume that

$$|\!|\!|[1,t]|\!|\!| \geqslant \frac{cn}{\alpha}\log\frac{|\Sigma|}{\alpha}, \qquad (100)$$

$$\operatorname{corr}[1,t] \leqslant \frac{1}{4} - \alpha, \qquad (101)$$

where $c \geqslant 1$ is the absolute constant from Lemma 41. Then as a result of the update, out is set to the leaf vertex in the unique root-to-leaf path in $X \cup Y$. An analogous theorem holds for Bob.

Proof: Analogous to the proof of Theorem 31 for a large alphabet, with the difference that the newly obtained Lemmas 34 and 41 should be used instead of their large-alphabet counterparts (Lemmas 20 and 30). ∎

We now establish the main result of this paper.

Proof of Theorem 32: The proof is nearly identical to that for a large alphabet (Theorem 17). Recall that $n = |\pi|$ denotes the communication cost of the original protocol, and $\epsilon > 0$ is a constant in the statement of Theorem 17. Consider the interactive coding scheme given by Algorithms 5 and 6 with parameters set according to

$$\alpha = \frac{\epsilon}{4}, \qquad (102)$$

$$N = \left\lceil \frac{cn}{2\alpha}\log\frac{|\Sigma|}{\alpha}\right\rceil + 1, \qquad (103)$$

where $c \geqslant 1$ is the absolute constant from Lemma 41. Then by (79), the interactive coding scheme uses an alphabet of size at most $2^{O(1/\epsilon^3)} = O(1)$. Furthermore, the combined number of transmissions sent by Alice and Bob does not exceed $2N = O(\frac{n}{\epsilon}\log\frac{|\Sigma|}{\epsilon}) = O(|\pi|\log|\Sigma|)$.

It remains to show that when the communication stops, *out* is set for both Alice and Bob to the leaf vertex on the unique root-to-leaf path in $X \cup Y$. To this end, recall from (102) and Lemma 28 that

$$\operatorname{corr}[1,T] \leqslant \frac{1}{4} - 2\alpha. \qquad (104)$$

By (103) and Lemma 25,

$$|\!|\!|[1,T]|\!|\!| \geqslant \frac{cn}{\alpha}\log\frac{|\Sigma|}{\alpha} + 2 \qquad (105)$$

and therefore

$$|\!|\!|[1,T-1]|\!|\!| \geqslant \frac{cn}{\alpha}\log\frac{|\Sigma|}{\alpha}. \qquad (106)$$

Also,

$$\begin{aligned}
\operatorname{corr}[1,T-1] &\leqslant \frac{|\!|\!|[1,T]|\!|\!|}{|\!|\!|[1,T-1]|\!|\!|}\cdot\operatorname{corr}[1,T]\\
&\leqslant \left(1 + \frac{2}{|\!|\!|[1,T-1]|\!|\!|}\right)\cdot\operatorname{corr}[1,T]\\
&\leqslant \left(1 + \frac{2\alpha}{n}\right)\cdot\left(\frac{1}{4} - 2\alpha\right)\\
&\leqslant \frac{1}{4} - \alpha, \qquad (107)
\end{aligned}$$

where the third step uses (104) and (106). Now, consider the last time that Alice and Bob update their copies of *out*. The complete sequence of symbols that Alice has received at the time of her last update is $\not\wr(R'[1,T-1])$ or $\not\wr(R'[1,T])$. Likewise, the complete sequence of symbols that Bob has received at the time of his last update is $\not\wr(R''[1,T-1])$ or $\not\wr(R''[1,T])$. By (104)–(107) and Theorem 42, both players set *out* to the leaf vertex in the unique root-to-leaf path in $X \cup Y$. ■

H. Generalization to early output

Following Braverman et al. [5], we now consider the setting when Alice and Bob need to be ready with their answers by a certain round (based on each player's own counting) rather than when the communication stops. Let Π be an interactive coding scheme. We define the *δ-early output* for a player in Π as the chronologically ordered sequence of the player's first $\delta|\Pi|/2$ symbols sent (or all of them, if the player sends fewer than $\delta|\Pi|/2$ symbols) and first $\delta|\Pi|/2$ symbols received (or all of them, if the player receives fewer than $\delta|\Pi|/2$ symbols). In this early output model, Alice and Bob are still expected to run their protocol to completion, which happens when one or both of them have finished $|\Pi|/2$ rounds of communication. The only difference is what information they use when computing their answers. Both Definition 10 and Theorem 32, item 1) generalize to the setting of early output, as follows.

Definition 43 (Coding scheme with early output): Let π be a given protocol with input space $\mathscr{X} \times \mathscr{Y}$. Protocol Π is an interactive coding scheme *for π with corruption rate ϵ and δ-early output* if:

1) Π has input space $\mathscr{X} \times \mathscr{Y}$ and is in canonical form;
2) there are functions f', f'' such that for any pair of inputs $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$ and any actions by an adversary with corruption rate ϵ, Alice's δ-early output a and Bob's δ-early output b satisfy $f'(a) = f''(b) = \pi(X,Y)$.

Theorem 44: Fix arbitrary constants $\epsilon > 0$ and $0 < \delta \leqslant 1$, and let π be an arbitrary protocol with alphabet Σ. Then there exists an interactive coding scheme for π with alphabet size $O(1)$ and communication cost $O(|\pi| \log|\Sigma|)$ with δ-early output that tolerates corruption rate $(\frac{1}{4}-\epsilon)\delta$.

Proof: Let $n = |\pi|$ denote the communication cost of the original protocol. Consider the interactive coding scheme given by Algorithms 5 and 6 with parameters set according to

$$\alpha = \frac{\epsilon}{2}, \tag{108}$$

$$N = \left\lceil \frac{cn}{\alpha\delta} \log \frac{|\Sigma|}{\alpha} + \frac{3}{\alpha\delta} \right\rceil, \tag{109}$$

where $c \geqslant 1$ is the absolute constant from Lemma 41. Then by (79), the interactive coding scheme uses an alphabet of size at most $2^{O(1/\epsilon^3)} = O(1)$. Furthermore, the combined number of transmissions sent by Alice and Bob does not exceed $2N = O(\frac{n}{\epsilon\delta} \log \frac{|\Sigma|}{\epsilon}) = O(|\pi| \log|\Sigma|)$.

It remains to show that each player's δ-early output uniquely determines the output of π. We will prove the following much stronger statement: at any point in time when *one* of the players has processed $\delta|\Pi|/2 = \delta N$ or more incoming symbols, the variable *out* is set for both Alice and Bob to the leaf vertex of the unique root-to-leaf path in $X \cup Y$. This will prove the theorem since one of the players is always guaranteed to be able to run the protocol to completion and in particular to receive $|\Pi|/2 = N$ symbols.

We now provide the details. Fix any integer $t \in \{1, 2, \ldots, T\}$ such that at least one of the players receives $\delta|\Pi|/2$ or more symbols over the course of transmissions $1, 2, \ldots, t$. This is equivalent to saying that $\max\{|G' \cup I'|_{[1,t]}, |G'' \cup I''|_{[1,t]}\} \geqslant \delta N$. Lemma 24 implies that

$$\|\![1,t]\|\! \geqslant 2\delta N - 1. \tag{110}$$

Now

$$\begin{aligned} \max\{\operatorname{corr}[1,t-1], \operatorname{corr}[1,t]\} &\leqslant \frac{|D|}{\|\![1,t-1]\|\!} \\ &\leqslant \frac{\left(\frac{1}{4}-\epsilon\right)\delta \cdot 2N}{2\delta N - 3} \\ &\leqslant \frac{1}{4} - \alpha, \end{aligned} \tag{111}$$

where the second step uses the bound $|D| \leqslant (\frac{1}{4}-\epsilon)\delta \cdot 2N$ in the hypothesis of the theorem, and the third step uses (108)–(110). Moreover, (109) and (110) ensure that

$$\|\![1,t]\|\! \geqslant \|\![1,t-1]\|\! \geqslant \frac{cn}{\alpha} \log \frac{|\Sigma|}{\alpha}. \tag{112}$$

Now, consider the last time that Alice and Bob update their copies of *out* over the course of transmissions $1, 2, \ldots, t$. The complete sequence of symbols that Alice has received at the time of her update is $\not\wr(R'[1,t-1])$ or $\not\wr(R'[1,t])$. Likewise, the complete sequence of symbols that Bob has received at the time of his update is $\not\wr(R''[1,t-1])$ or $\not\wr(R''[1,t])$. By (111), (112), and Theorem 42, both players set *out* to the leaf vertex in the unique root-to-leaf path in $X \cup Y$. ■

In the terminology of our paper, Braverman et al. [5] studied interactive coding schemes with $(1-2\eta)$-early output that tolerate corruption rate η. As their main result, they proved the existence of such a scheme with alphabet size $O(1)$ and communication cost $O(|\pi| \log|\Sigma|)$ for any constant $\eta < 1/18$. They also showed that no such scheme exists in general for $\eta \geqslant 1/6$. Our paper closes the gap between the $1/18$ and $1/6$, establishing the existence of an interactive coding scheme for any π and any constant $\eta < 1/6$. This can be seen by taking $\delta = 1 - 2\eta$ and $\epsilon = \frac{1}{4} - \frac{\eta}{1-2\eta}$ in Theorem 44.

I. Optimality

We now establish the optimality of Theorem 32, showing that it tolerates the highest possible corruption rate and normalized corruption rates. We do so by studying the *pointer jumping protocol* PJP_n, defined for $n \geqslant 1$ as the protocol with input space $\{0,1\}^n \times \{0,1\}^n$ in which Alice and Bob exchange their strings one bit at a time, taking turns after every bit. Thus, the sequence of symbols exchanged on input (x,y) is $x_1 y_1 \ldots x_n y_n$. We show that no interactive coding scheme with alphabet size $2^{o(n)}$ for PJP_n can tolerate a corruption rate, or normalized corruption rates, higher than those tolerated by Theorem 32 with a constant-size alphabet. Our proof uses the "cut and paste" technique of previous impossibility results [6], [5]. We will first establish a detailed technical theorem and then deduce our impossibility results as corollaries.

Theorem 45: Let $\epsilon_{\text{subs}}, \epsilon_{\text{oos}} \geqslant 0$ be given. Suppose that Π is an interactive coding scheme with alphabet Σ for PJP_n that tolerates normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$. Then

$$\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} < \frac{1}{4} + \frac{\log|\Sigma|}{n}. \tag{113}$$

Proof: Let $N = |\Pi|/2$ be the number of communication rounds in Π. Since Π simulates PJP_n, the former produces at least as many distinct transcripts as the latter. This leads to $|\Sigma|^{2N} \geqslant 4^n$ and

$$N \geqslant \frac{n}{\log|\Sigma|}. \tag{114}$$

The centerpiece of the proof is the following claim.

Claim 46: Assume the hypothesis of Theorem 45, so that Π is an interactive coding scheme with alphabet Σ for PJP_n that tolerates normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$. Then the system

$$k \leqslant \epsilon_{\text{oos}}(2N - k), \tag{115}$$

$$\left\lceil \frac{N}{2} \right\rceil - k \leqslant \epsilon_{\text{subs}}(2N - k) \tag{116}$$

has no integral solution $0 \leqslant k \leqslant \lceil N/2 \rceil$.

Proof: For the sake of contradiction, suppose that the system has a solution $k \in \{0, 1, 2, \ldots, \lceil N/2 \rceil\}$. Fix arbitrary $x, y, y' \in \{0,1\}^n$ with $y \neq y'$, and consider the following two executions of Π.

1) Alice and Bob receive inputs x and y, respectively. The adversary uses substitution attacks to replace Bob's first $\lceil N/2 \rceil - k$ responses to Alice with the corresponding responses that he would send if his input were y'. Then the adversary carries out k consecutive out-of-sync attacks, intercepting Alice's transmissions to Bob and sending back to Alice the responses that Bob would send at that point if his input were y'. From then on, the adversary does not interfere with the communication. We let $\sigma_1, \sigma_2, \ldots, \sigma_N \in \Sigma$ denote the complete sequence of symbols that Alice receives in this execution.

2) Alice and Bob receive inputs x and y', respectively. The adversary does not interfere with the first $\lfloor N/2 \rfloor$ rounds of communication. As a result, the sequence of symbols that Alice receives in those rounds is $\sigma_1, \sigma_2, \ldots, \sigma_{\lfloor N/2 \rfloor}$. The adversary tampers with every symbol delivered to Alice from then on, making sure that she receives the sequence $\sigma_{\lfloor N/2 \rfloor + 1}, \ldots, \sigma_{N-1}, \sigma_N$. The adversary does so using $\lceil N/2 \rceil - k$ consecutive substitution attacks followed by k consecutive out-of-sync attacks. At that point, the communication stops because Alice has received N symbols.

Both executions feature $2N - k$ transmissions, $\lceil N/2 \rceil - k$ substitution attacks, and k out-of-sync attacks. By (115) and (116), these numbers of substitution and out-of-sync attacks are legitimate under normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$. As a result, Alice and Bob's simulation of PJP_n is correct in both executions. Since PJP_n produces different transcripts on (x, y) and (x, y'), we conclude that both Alice and Bob are able to distinguish between the two executions. We have reached the promised contradiction because the two executions look identical to Alice. ∎

We now return to the proof of the theorem. The values $k \in [0, \lceil N/2 \rceil]$ that satisfy (115) form a subinterval of $[0, \lceil N/2 \rceil]$ that contains 0. Analogously, the values $k \in [0, \lceil N/2 \rceil]$ that satisfy (116) form a subinterval of $[0, \lceil N/2 \rceil]$ that contains $\lceil N/2 \rceil$. Since the system of these two inequalities has no integral solution in $[0, \lceil N/2 \rceil]$, there exists $k^* \in [0, \lceil N/2 \rceil - 1]$ such that $k = k^* + 1$ and $k = k^*$ violate (115) and (116), respectively:

$$\epsilon_{\text{oos}} < \frac{k^* + 1}{2N - k^* - 1},$$
$$\epsilon_{\text{subs}} < \frac{\lceil N/2 \rceil - k^*}{2N - k^*}.$$

Taking a weighted sum of these inequalities with weights $3/4$ and 1,

$$\begin{aligned}\frac{3}{4}\epsilon_{\text{oos}} + \epsilon_{\text{subs}} &< \frac{3}{4} \cdot \frac{k^* + 1}{2N - k^* - 1} + \frac{\frac{1}{2}(N+1) - k^*}{2N - k^*} \\ &= \frac{1}{4} + \frac{5N - k^* - 1}{2(2N - k^* - 1)(2N - k^*)} \\ &\leqslant \frac{1}{4} + \frac{1}{N},\end{aligned}$$

where the last step uses $k^* \leqslant (N-1)/2$. By (114), the proof is complete. ∎

We now derive our claimed impossibility results as corollaries of Theorem 45.

Corollary 47: Suppose that for every $n \geqslant 1$, there is an interactive coding scheme for the pointer jumping protocol PJP_n with alphabet size $2^{o(n)}$ that tolerates normalized corruption rate $(\epsilon_{\text{subs}}, \epsilon_{\text{oos}})$. Then

$$\epsilon_{\text{subs}} + \frac{3}{4}\epsilon_{\text{oos}} \leqslant \frac{1}{4}.$$

Proof: Substitute $|\Sigma| = 2^{o(n)}$ in Theorem 45 and pass to the limit as $n \to \infty$. ∎

Corollary 48: Suppose that for every $n \geqslant 1$, there is an interactive coding scheme for the pointer jumping protocol PJP_n with alphabet size $2^{o(n)}$ that tolerates corruption rate ϵ. Then

$$\epsilon \leqslant \frac{1}{4}.$$

Proof: Any scheme that tolerates corruption rate ϵ must also tolerate normalized corruption rate $(\epsilon, 0)$. Therefore, the claim follows by taking $\epsilon_{\text{subs}} = \epsilon$ and $\epsilon_{\text{oos}} = 0$ in Corollary 47. ∎

Acknowledgment

The authors are thankful to Mark Braverman, Rafail Ostrovsky, and anonymous reviewers for their valuable feedback on an earlier version of this manuscript.

Appendix

The purpose of this appendix is to prove Theorem 9 on the existence of α-good codes, which we now restate for the reader's convenience.

Theorem 49 (restatement of Theorem 9): For any alphabet Σ_{in}, any $0 < \alpha < 1$, and any integer $n \geqslant 0$, there is an α-good code $C\colon \Sigma_{\text{in}}^{\leqslant n} \to \Sigma_{\text{out}}^{\leqslant n}$ of depth n with

$$|\Sigma_{\text{out}}| = \left\lceil \frac{(10|\Sigma_{\text{in}}|)^{1/\alpha}\,\mathrm{e}}{\alpha} \right\rceil^2 . \tag{117}$$

Our treatment is a reworked and simplified version of an argument of Braverman et al. [5], who proved the existence of a closely related family of tree codes.

We fix α for the rest of the proof and define Σ_{out} to be the alphabet of consecutive natural numbers, with cardinality given by (117). For strings u and v, we write $u \diamond v$ to mean that $\mathrm{ED}(u,v) < (1-\alpha)(|u|+|v|)$. For a tree code $C\colon \Sigma_{\text{in}}^{\leqslant n} \to \Sigma_{\text{out}}^{\leqslant n}$ of depth n and a string $u \in \Sigma_{\text{in}}^{\leqslant n}$, we let C_u denote the tree code $C_u\colon \Sigma_{\text{in}}^{\leqslant n-|u|} \to \Sigma_{\text{out}}^{\leqslant n-|u|}$ of depth $n-|u|$ given by $C_u(v) = (C(uv))_{>|u|}$.

Our proof centers around two inductively defined families $\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_n, \ldots$ and $\mathscr{C}_0^*, \mathscr{C}_1^*, \mathscr{C}_2^*, \ldots, \mathscr{C}_n^*, \ldots$, where $\mathscr{C}_n$ and $\mathscr{C}_n^*$ are sets of tree codes of depth n. As a base case, we let $\mathscr{C}_0 = \mathscr{C}_0^*$ be the family whose only member is the tree code $\varepsilon \mapsto \varepsilon$, which is by definition the only tree code of depth 0. Assuming that $\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_{n-1}$ and $\mathscr{C}_0^*, \mathscr{C}_1^*, \mathscr{C}_2^*, \ldots, \mathscr{C}_{n-1}^*$ have been constructed, we define $\mathscr{C}_n$ to be the family of all tree codes $C\colon \Sigma_{\text{in}}^{\leqslant n} \to \Sigma_{\text{out}}^{\leqslant n}$ of depth n such that $C_\sigma \in \mathscr{C}_{n-1}^*$ for all $\sigma \in \Sigma_{\text{in}}$, and define $\mathscr{C}_n^*$ to be the family of all α-good codes in $\mathscr{C}_n$. To settle Theorem 9, it remains to prove that each $\mathscr{C}_n^*$ is nonempty. We will in fact prove the following stronger claim:

$$\frac{|\mathscr{C}_n^*|}{|\mathscr{C}_n|} \geqslant \frac{1}{2}, \qquad n = 0,1,2,3,\ldots. \tag{118}$$

We will argue by induction on n. The base case $n = 0$ is trivial. For the inductive step, fix $n \geqslant 1$ arbitrarily and assume that $|\mathscr{C}_i^*|/|\mathscr{C}_i| \geqslant 1/2$ for $i = 0,1,2,\ldots,n-1$. A key technical element of our analysis is the following observation.

Claim 50: Let $u, v, w \in \Sigma_{\text{in}}^*$ be given, where

$$v_{\leqslant 1} \neq w_{\leqslant 1},$$
$$|uv| \leqslant n,$$
$$|uw| \leqslant n.$$

Then

$$\mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C(uv) \diamond C_u(w)] \leqslant \left(\frac{1}{5|\Sigma_{\text{in}}|}\right)^{|u|+|v|+|w|}.$$

The hypothesis $v_{\leqslant 1} \neq w_{\leqslant 1}$ above amounts to saying that the longest common prefix of v and w is the empty string.

Proof of Claim 50: The claim follows from the following derivation, whose steps we will justify shortly:

$$\begin{aligned}
&\mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C(uv) \diamond C_u(w)] \\
&\qquad \leqslant 2^{|u|} \mathop{\mathbf{P}}_{\substack{z \in \Sigma_{\text{out}}^{|u|} \\ C \in \mathscr{C}_{n-|u|}}} [zC(v) \diamond C(w)] && (119)\\
&\qquad = 2^{|u|} \mathop{\mathbf{P}}_{\substack{z \in \Sigma_{\text{out}}^{|u|} \\ C', C'' \in \mathscr{C}_{n-|u|}}} [zC'(v) \diamond C''(w)] && (120)\\
&\qquad \leqslant 2^{|u|+|v|} \mathop{\mathbf{P}}_{\substack{z \in \Sigma_{\text{out}}^{|u|} \\ z' \in \Sigma_{\text{out}}^{|v|} \\ C'' \in \mathscr{C}_{n-|u|}}} [zz' \diamond C''(w)] && (121)\\
&\qquad \leqslant 2^{|u|+|v|+|w|} \mathop{\mathbf{P}}_{\substack{z \in \Sigma_{\text{out}}^{|u|} \\ z' \in \Sigma_{\text{out}}^{|v|} \\ z'' \in \Sigma_{\text{out}}^{|w|}}} [zz' \diamond z''] && (122)\\
&\qquad \leqslant 2^{|u|+|v|+|w|} \left(\frac{\mathrm{e}}{\alpha\sqrt{|\Sigma_{\text{out}}|}}\right)^{\alpha(|u|+|v|+|w|)} && (123)\\
&\qquad \leqslant \frac{1}{(5|\Sigma_{\text{in}}|)^{|u|+|v|+|w|}}. && (124)
\end{aligned}$$

Inequality (119) is trivially true for $u = \varepsilon$. To verify validity for $|u| \geqslant 1$, observe that

$$\begin{aligned}
&\mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C(uv) \diamond C_u(w)] \\
&\quad = \mathop{\mathbf{P}}_{\substack{z_1 \in \Sigma_{\text{out}} \\ C \in \mathscr{C}_{n-1}^*}} [z_1 C(u_{\geqslant 2} v) \diamond C_{u_{\geqslant 2}}(w)] \\
&\quad \leqslant \mathop{\mathbf{P}}_{\substack{z_1 \in \Sigma_{\text{out}} \\ C \in \mathscr{C}_{n-1}}} [z_1 C(u_{\geqslant 2} v) \diamond C_{u_{\geqslant 2}}(w)] \cdot \frac{|\mathscr{C}_{n-1}|}{|\mathscr{C}_{n-1}^*|} \\
&\quad \leqslant \mathop{\mathbf{P}}_{\substack{z_1 \in \Sigma_{\text{out}} \\ C \in \mathscr{C}_{n-1}}} [z_1 C(u_{\geqslant 2} v) \diamond C_{u_{\geqslant 2}}(w)] \cdot 2,
\end{aligned}$$

where the first step uses the definition of $\mathscr{C}_n$, and the last two steps use $\mathscr{C}_{n-1}^* \subseteq \mathscr{C}_{n-1}$ and $|\mathscr{C}_{n-1}^*| \geqslant |\mathscr{C}_{n-1}|/2$. Applying this maneuver an additional $|u|-1$ times settles (119). The next step, (120), is valid because the longest common prefix of v and w is the empty string and therefore $C(v)$ and $C(w)$ are independent. Steps (121) and (122) can be verified in a manner identical to (119). The final steps (123) and (124) follow from Proposition 4 and (117), respectively. ■

Armed with Claim 50, we are now in a position to complete the inductive step. Our objective is to show that $|\mathscr{C}_n^*|/|\mathscr{C}_n| \geqslant 1/2$, or equivalently that a uniformly random code $C \in \mathscr{C}_n$ has an α-violation with probability at most $1/2$. Recall that an α-violation in C is a quadruple of vertices (A, B, D, E) in the tree representation of C with the following properties:

1) B is the deepest common predecessor of D and E;
2) A is a predecessor of B;
3) $AD \diamond BE$, where $AD \in \Sigma_{\text{out}}^*$ and $BE \in \Sigma_{\text{out}}^*$ denote the concatenation of the code symbols along the path from A to D and the path from B to E, respectively.

We further deduce that

4) A is the root;

5) $B \neq E$.

The former holds because the codes in $\mathscr{C}^*_{n-1}$ have no α-violations, and the latter follows from Remark 7. These structural constraints allow us to identify an α-violation (A, B, D, E) in C in a one-to-one manner with a triple of strings $u, v, w \in \Sigma^*_{\text{in}}$ such that $v_{\leqslant 1} \neq w_{\leqslant 1}$, $w \neq \varepsilon$, and $C(uv) \diamond C_u(w)$. Applying the union bound over all such triples u, v, w,

$$\mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C \text{ has an } \alpha\text{-violation}] \leqslant \sum_{\substack{u \in \Sigma^*_{\text{in}}: \\ |u| < n}} \sum_{\substack{v \in \Sigma^*_{\text{in}}: \\ |v| \leqslant n - |u|}} \sum_{\substack{w \in \Sigma^+_{\text{in}}: \\ |w| \leqslant n - |u|, \\ w_{\leqslant 1} \neq v_{\leqslant 1}}} \mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C(uv) \diamond C_u(w)].$$

Appealing to Claim 50 and simplifying,

$$\begin{aligned}
&\mathop{\mathbf{P}}_{C \in \mathscr{C}_n}[C \text{ has an } \alpha\text{-violation}] \\
&\quad \leqslant \sum_{\substack{u \in \Sigma^*_{\text{in}}: \\ |u| < n}} \sum_{\substack{v \in \Sigma^*_{\text{in}}: \\ |v| \leqslant n - |u|}} \sum_{\substack{w \in \Sigma^+_{\text{in}}: \\ |w| \leqslant n - |u|, \\ w_{\leqslant 1} \neq v_{\leqslant 1}}} \left(\frac{1}{5|\Sigma_{\text{in}}|} \right)^{|u|+|v|+|w|} \\
&\quad \leqslant \sum_{u \in \Sigma^*_{\text{in}}} \sum_{v \in \Sigma^*_{\text{in}}} \sum_{w \in \Sigma^+_{\text{in}}} \left(\frac{1}{5|\Sigma_{\text{in}}|} \right)^{|u|+|v|+|w|} \\
&\quad = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} \frac{1}{5^{i+j+k}} \\
&\quad = \frac{1}{5} \cdot \frac{1}{\left(1 - \frac{1}{5}\right)^3} \\
&\quad < \frac{1}{2}.
\end{aligned}$$

The final inequality is equivalent to $|\mathscr{C}^*_n|/|\mathscr{C}_n| > 1/2$, completing the inductive step. We have settled (118) and thereby proved Theorem 9.

References

[1] S. Agrawal, R. Gelles, and A. Sahai, "Adaptive protocols for interactive communication," in *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, 2016, pp. 595–599.

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

[3] G. Brassard, A. Nayak, A. Tapp, D. Touchette, and F. Unger, "Noisy interactive quantum communication," in *Proceedings of the Fifty-Fifth Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2014, pp. 296–305.

[4] M. Braverman and K. Efremenko, "List and unique coding for interactive communication in the presence of adversarial noise," in *Proceedings of the Fifty-Fifth Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2014, pp. 236–245.

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

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

[7] K. Efremenko, R. Gelles, and B. Haeupler, "Maximal noise in interactive communication over erasure channels and channels with feedback," *IEEE Trans. Information Theory*, vol. 62, no. 8, pp. 4575–4588, 2016.

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

[9] R. Gelles, "Coding for interactive communication: A survey," *Foundations and Trends in Theoretical Computer Science*, vol. 13, no. 1-2, pp. 1–157, 2017.

[10] R. Gelles and B. Haeupler, "Capacity of interactive communication over erasure channels and channels with feedback," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, 2015, pp. 1296–1311.

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

[12] M. Ghaffari and B. Haeupler, "Optimal error rates for interactive coding II: efficiency and list decoding," in *Proceedings of the Fifty-Fifth Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2014, pp. 394–403.

[13] M. Ghaffari, B. Haeupler, and M. Sudan, "Optimal error rates for interactive coding I: adaptivity and other settings," in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing* (STOC), 2014, pp. 794–803.

[14] B. Haeupler, "Interactive channel capacity revisited," in *Proceedings of the Fifty-Fifth Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2014, pp. 226–235.

[15] S. Jukna, *Extremal Combinatorics with Applications in Computer Science*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2011.

[16] J. Justesen, "Class of constructive asymptotically good algebraic codes," *IEEE Trans. Information Theory*, vol. 18, no. 5, pp. 652–656, 1972.

[17] G. Kol and R. Raz, "Interactive channel capacity," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (STOC), 2013, pp. 715–724.

[18] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[19] R. Ostrovsky, Y. Rabani, and L. J. Schulman, "Error-correcting codes for automatic control," *IEEE Trans. Information Theory*, vol. 55, no. 7, pp. 2931–2941, 2009.

[20] L. J. Schulman, "Communication on noisy channels: A coding theorem for computation," in *Proceedings of the Thirty-Third Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 1992, pp. 724–733.

[21] ——, "Deterministic coding for interactive communication," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* (STOC), 1993, pp. 747–756.

[22] ——, "Coding for interactive communication," *IEEE Trans. Information Theory*, vol. 42, no. 6, pp. 1745–1756, 1996.

[23] L. J. Schulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," *IEEE Trans. Information Theory*, vol. 45, no. 7, pp. 2552–2557, 1999.

[24] A. C.-C. Yao, "Some complexity questions related to distributive computing," in *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing* (STOC), 1979, pp. 209–213.