58th Annual IEEE Symposium on Foundations of Computer Science

Optimal Interactive Coding for Insertions, Deletions, and Substitutions

Alexander A. Sherstov and Pei Wu
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095 USA
Email: {sherstov, pwu}@cs.ucla.edu

***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. They showed how to faithfully simulate any protocol in this model with corruption rate up to 1/18, 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 while keeping the alphabet to a constant size and the communication overhead to a constant factor. Our corruption tolerance matches an impossibility result for corruption rate 1/4 which holds even for substitutions alone (Braverman and Rao, STOC '11).

***Keywords*-interactive coding; communication complexity; tree codes; edit distance; insertions and deletions**

I. Introduction

Shannon [1], [2] famously considered the problem of transmitting a message over an unreliable channel. 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. Shannon's problem is the subject matter of coding theory and has been extensively studied. In particular, for any constant $\epsilon > 0$, it is known [3] 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 [4]–[6] considered a generalization of Shannon's 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. 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 Shannon's 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 [6] 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, from corruption rate to communication overhead to computational complexity. We refer the reader to Gelles [7] for an up-to-date survey. Of particular interest to us is the work of Braverman and Rao [8], 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 [6], their simulation 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 [9]

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.30

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 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 [10] 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 [3]. As their main result, Braverman et al. [9] 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.

The authors of [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. [9] 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. [9] 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. The adversary is free to decide which party will receive the inserted symbol. This makes possible two types of attacks. 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 [6]. 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. [9] 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.

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 [8], which holds even if the adversary is restricted to substitutions.

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 the following theorem.

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; see the full version of this paper [11, Section 5.9]. 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 the full version of this paper [11, Section 5.8], we further 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. [9] 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 [9, Theorem G.1].

C. Background on interactive coding

In what follows, we review relevant previous work [6], [8], [9] 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 [6]. 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 [6] 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 1 (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.

D. The Braverman–Rao simulation

We are now in a position to describe the simulation of Braverman and Rao [8] 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

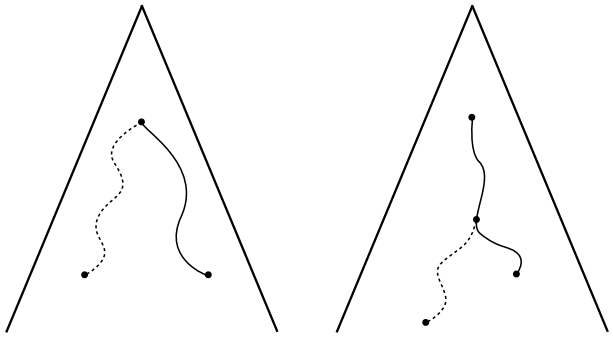

Figure 1. Distance constraints for codewords in a tree code (left) and an edit distance tree code (right).

of outgoing edges for the vertices at odd depths. Execution of π, then, corresponds to identifying the unique root-to-leaf 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. 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 [8] 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. [9] in the BGMO model with substitutions, insertions, and deletions. The authors of [9] draw inspiration from the classic work of Levenshtein [12], 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 [3]. 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. They authors of [9] 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 1 (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 [8], 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 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. [9] 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 $\max\{\ell_1, \ell_2\}$; see the full version of this paper [11, Prop. 2.2]. 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

A. General

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. 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 $0/0 = 0$. We let $\log x$ denote the logarithm of x to base 2.

B. String notation

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 $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. 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 relations $\succ, \preceq, \succeq$ are defined analogously. 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, we use the subsequence notation

$v|_A = v_{i_1} v_{i_2} \dots v_{i_{|A|}}$, 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.

C. Edit distance

Recall that the asterisk $*$ is a reserved symbol that does not appear in any alphabet Σ in this paper. 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. Thus, $*(v)+\overline{*}(v) = |v|$. 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:

(i) $|S| = |R|$,
(ii) $\not*(S) = s$,
(iii) $\not*(R) = r$,
(iv) $\forall i: \quad R_i \neq * \ \vee \ S_i \neq *$,
(v) $\forall i: \quad (R_i \neq * \ \wedge \ S_i \neq *) \implies R_i = S_i$.

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 *edit distance* between strings $s, r \in \Sigma^*$ is denoted $\mathrm{ED}(s, r)$ and is given by $\mathrm{ED}(s, r) = \min\{*(S) + *(R)\}$, where the minimum is over all alignments $S \parallel R$ for s, r. Equivalently, $\mathrm{ED}(s, r)$ 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 ($S_i = R_i \neq *$).

D. Suffix distance

For an alignment $S \parallel R$, define $\Delta(S, R) = (*(S) + *(R))/\overline{*}(S)$. 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 *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. [9], inspired in turn by an earlier notion of suffix distance due to Schulman [6]. 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

$$0 = \mathrm{SD}_0 \leqslant \mathrm{SD}_1 \leqslant \mathrm{SD}_2 \leqslant \mathrm{SD}_3 \leqslant \cdots \leqslant \mathrm{SD}_\infty = \mathrm{SD}\,. \qquad (1)$$

We generalize the above definitions to strings $s, r \in \Sigma^*$ by letting $\mathrm{SD}(s, r) = \min \mathrm{SD}(S, R)$ and $\mathrm{SD}_k(s, r) = \min \mathrm{SD}_k(S, R)$, where in both cases the minimum is over all alignments $S \parallel R$ for s, r. The proof of the following proposition can be found in the full version of our paper [11, Prop. 3.1].

Proposition 3. *Fix alignments $S' \parallel R'$ and $S'' \parallel R''$. Then:*

(i) $\Delta(S'S'', R'R'') \leqslant \max\{\Delta(S', R'), \Delta(S'', R'')\}$;
(ii) $\Delta(S'S'', R'R'') \geqslant \min\{\Delta(S', R'), \Delta(S'', R'')\}$;
(iii) $\mathrm{SD}(S'S'', R'R'') \leqslant \max\{\mathrm{SD}(S', R'), \mathrm{SD}(S'', R'')\}$;
(iv) $\mathrm{SD}_k(S'S'', R'R'') \leqslant \max\{\mathrm{SD}_k(S', R'), \Delta(S'', R'')\}$ *for* $k \leqslant \overline{*}(S')$.

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.

Definition 4 (α-violation). Fix a tree code $C\colon \Sigma_{\mathrm{in}}^* \to \Sigma_{\mathrm{out}}^*$ 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:

(i) B is the deepest common predecessor of D and E;
(ii) A is any predecessor of B; and
(iii) $\mathrm{ED}(AD, BE) < (1 - \alpha)(|AD| + |BE|)$, where $AD \in \Sigma_{\mathrm{out}}^*$ is the concatenation of the code symbols along the path from A to D, and analogously $BE \in \Sigma_{\mathrm{out}}^*$ is the 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 4 strengthens an earlier formalism due to Braverman et al. [9], in which the inequality $\mathrm{ED}(AD, BE) < (1-\alpha)\max\{|AD|, |BE|\}$ played the role of our constraint (iii). The strengthening is essential to the tight results of our paper. The following theorem ensures the existence of α-good codes with good parameters.

Theorem 5. *For any alphabet Σ_{in}, any $0 < \alpha < 1$, and any integer $n \geqslant 0$, there is an α-good code $C\colon \Sigma_{\mathrm{in}}^* \to \Sigma_{\mathrm{out}}^*$ of depth n with*

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

This theorem and its proof are adaptations of an earlier result due to Braverman et al. [9]. A complete and self-contained

proof of Theorem 5 is available in the full version of this paper [11, Appendix A].

FACT 6. *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.*

A proof of this fact is available in the full version of this paper [11, Fact 2.6].

In contrast to the work of Braverman et al. [9], 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. Our decoding is based on the following theorem, proved in the full version of this paper [11, Thm. 3.4].

THEOREM 7. *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* $(\textsc{Decode}_{C,\alpha}(r))_{\leqslant k} = s_{\leqslant k}$ *for any real* $0 \leqslant k \leqslant \infty$, *any codeword* s, *and any string* $r \in \Sigma_{\text{out}}^*$ *with* $\text{SD}_k(s, r) < 1 - \alpha$.

F. Communication model

In a communication protocol, the transfer of an alphabet symbol from one party to the other is an atomic operation to which we refer as a *transmission*. 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 π, denoted $|\pi|$, is the worst-case number of transmissions, or equivalently the maximum length of the protocol output on any input pair.

We adopt the corruption model due to Braverman et al. [9], reviewed in the introduction. With the adversary present, the *output* of a player in a particular execution is the complete sequence of symbols, ordered chronologically, that that player sends and receives over the course of the execution.

DEFINITION 8 (Coding scheme). Let π be a given protocol with input space $\mathcal{X} \times \mathcal{Y}$. We say that protocol Π is an *interactive coding scheme for* π *that tolerates corruption rate* ϵ if:

(i) Π has input space $\mathcal{X} \times \mathcal{Y}$ and is in canonical form;
(ii) when Π is executed on a given pair of inputs $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, 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;
(iii) there exist functions f', f'' such that for any pair of inputs $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ and any allowable behavior by the adversary, Alice's output a and Bob's output b satisfy $f'(a) = f''(b) = \pi(X, Y)$.

Coding schemes for *normalized* corruption rate are defined analogously.

III. 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:

THEOREM 9. *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*

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

We have organized our proof around nine milestones, corresponding to Sections III-A–III-I. Looking ahead, we will obtain the main result of this paper by improving the alphabet size to a constant.

A. The simulation

Recall 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 [8], [9], 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; see the full version [11, Lem. 4.2] for the proof.

LEMMA 10. *Consider an arbitrary point in time, and let*

$$(i_1, \sigma_1), (i_2, \sigma_2), \ldots (i_t, \sigma_t) \tag{2}$$

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

A formal description of our interactive coding scheme is given in Figures 2 and 3 for Alice and Bob, respectively.

Input: X (set of Alice's edges)

1 encode and send the edge in X incident to the root

2 **for** $i = 1, 2, 3, \ldots, N$ **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 ℓ, 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 if**

12 **end for**

Figure 2. The coding scheme for Alice.

Input: Y (set of Bob's edges)

1 **for** $i = 1, 2, 3, \ldots, N$ **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 ℓ, 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 for**

Figure 3. The coding scheme for Bob.

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 5. 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 7. 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 5,

$$|\Sigma_{\text{out}}| = (|\Sigma| \cdot N)^{O(1/\alpha)}. \tag{3}$$

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 6 corresponds to a unique string in Σ_{in}^*. Recall that this string is of the form (2) 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 10 to convert (2) to a sequence of edges. It may happen that (2) is syntactically malformed; in that case, the receiving party interrupts the interpretation process at the longest prefix of (2) 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 III-B–III-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 (i), (ii) in Theorem 9. 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 9.

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.

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''. 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

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

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 G'', D'', I'' for Bob, and abbreviate

$$G = G' \cup G'', \qquad D = D' \cup D'', \qquad I = I' \cup I''.$$

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

$$T \leqslant 2N. \tag{4}$$

C. Excellent transmissions

As we now show, the codewords $\not\in(S'[1,t])$ and $\not\in(S''[1,t])$ completely reveal the sequences of edges sent by Bob and by Alice, respectively, over the course of the first t transmissions.

Lemma 11. *Let $t \in \{1,2,\dots,T\}$ be given. Then:*

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

The proof of this lemma is available in the full version of the paper [11, Lem. 4.4]. 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 12. *Let $t \in \{1,2,\dots,T\}$ be given.*

(i) *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.*
(ii) *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 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 7, we conclude that $\textsc{Decode}_{C,\alpha}(\not\in(R'[1,t])) = \not\in(S'[1,t])$. This means that on receipt of transmission t, Alice is able to correctly recover the entire codeword $\not\in(S'[1,t])$ that Bob has sent her so far. By Lemma 11, 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 III-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 $B' = (G' \cup I') \backslash E'$ and $B'' = (G'' \cup I'') \backslash E''$, respectively. We abbreviate $B = B' \cup B''$. 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, proved in the full version [11, Lem. 4.7].

Lemma 13. *For any interval J with $1 \in J$,*

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

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

$$\|Z\| = |G' \cup I' \cup D'|_Z + |G'' \cup I'' \cup D''|_Z.$$

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 $Z \subseteq \{1,2,\dots,T\}$, and a moment's thought reveals that the lower and upper bounds can both be attained. The next three lemmas establish key facts about virtual length; see the full version [11, Lems. 4.9–4.11] for the proofs.

Lemma 14. *The total virtual length of all transmissions satisfies $\|[1,T]\| \geqslant 2N$.*

Lemma 15. *Let i, j be given integers with $i \leqslant j$. Then*

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

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

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

Lemma 16. *For any interval J,*

$$\|J\| \leqslant 2(|B|_J + |E'|_J) + 1,$$
$$\|J\| \leqslant 2(|B|_J + |E''|_J) + 1.$$

F. Virtual corruption rate

In keeping with our focus on events rather than transmissions, we define $\operatorname{corr} J = |D \cap J|/\|J\|$ 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 [13]. 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; see the full version [11, Lem. 4.12] for the proof.

Lemma 17. *Assumptions* (i) *and* (ii) *in Theorem* 9 *imply* $\operatorname{corr}[1,T] \leqslant \frac{1}{4} - \epsilon$ *and* $\operatorname{corr}[1,T] \leqslant \frac{1}{4} - \frac{\epsilon}{2}$, *respectively.*

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

$$[\overline{f_{i-1}}, \overline{f_i}) = [\overline{f_{i-1}}, f_i), \qquad i = 1, 2, \ldots, n. \tag{5}$$

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 18. *For any integers i and t with* $\overline{f_{i-1}} \leqslant t < f_i$,

$$\|[\overline{f_{i-1}}, t]\| \leqslant 2|B|_{[\overline{f_{i-1}}, t]} + 3. \tag{6}$$

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 12 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 (6) in light of Lemma 16. ∎

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

$$\|[1,t]\| \geqslant \frac{n+2}{\alpha}, \tag{7}$$
$$\operatorname{corr}[1,t] \leqslant \frac{1}{4} - \alpha. \tag{8}$$

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

$$[\overline{f_n}, t^*) \cap E' \neq \varnothing, \tag{9}$$
$$[\overline{f_n}, t^*) \cap E'' \neq \varnothing, \tag{10}$$
$$\Delta(S'[i,t], R'[i,t]) < 1 - \alpha, \qquad i = 1, 2, \ldots, t^*, \tag{11}$$
$$\Delta(S''[i,t], R''[i,t]) < 1 - \alpha, \qquad i = 1, 2, \ldots, t^*. \tag{12}$$

Proof. Equations (11) and (12) 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 (11) and (12) hold. For the sake of contradiction, assume that at least one of the remaining desiderata (9), (10) is violated, whence

$$\|[\overline{f_n}, t^*)\| \leqslant 2|B|_{[\overline{f_n}, t^*)} + 1 \tag{13}$$

by Lemma 16. The proof strategy is to show that (13) 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,

$$\begin{aligned} \|[\overline{f_{i-1}}, \overline{f_i})\| &= \|[\overline{f_{i-1}}, f_i)\| \\ &\leqslant 2|B|_{[\overline{f_{i-1}}, f_i)} + 3 \\ &\leqslant 2|B|_{[\overline{f_{i-1}}, \overline{f_i})} + 3 \end{aligned} \tag{14}$$

for any $i = 1, 2, \ldots, n^*$, where the first and third steps use (5), and the second step follows from Lemma 18. Next, the upper bound

$$\|[\overline{f_{n^*}}, t^*)\| \leqslant 2|B|_{[\overline{f_{n^*}}, t^*)} + 3 \tag{15}$$

follows from Lemma 18 if $n^* < n$ and from (13) if $n^* = n$. The virtual length of the singleton interval $\{t^*\}$ can be bounded from first principles:

$$\|\{t^*\}\| \leqslant 2. \tag{16}$$

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

$$\|(t^*, t]\| \leqslant \frac{4}{1-\alpha}|D|_{(t^*,t]} + 1 \tag{17}$$

by Lemma 15 in the former case and trivially in the latter.

Putting everything together, we obtain

$$\begin{aligned}
\|[1, t]\| &\leqslant 2|B|_{[0,t^*)} + 3(n^* + 1) + 2 + \frac{4}{1-\alpha}|D|_{(t^*,t]} + 1 \\
&\leqslant \frac{4}{1-\alpha}|D|_{[0,t^*)} + 3(n^* + 1) + 2 + \frac{4}{1-\alpha}|D|_{(t^*,t]} + 1 \\
&\leqslant \frac{4}{1-\alpha}|D|_{[0,t]} + 3n + 6 \\
&\leqslant \frac{4}{1-\alpha}|D|_{[0,t]} + 3\alpha\|[1, t]\|,
\end{aligned} \tag{18}$$

where the first step is the result of adding (14)–(17), the second step applies Lemma 13, and the final step uses (7). Since $0 < \alpha < 1$, the conclusion of (18) is equivalent to

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

which is inconsistent with (8). We have obtained the desired contradiction and thereby proved that t^* satisfies each of the properties (9)–(12). ∎

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 9.

Theorem 20. *Consider a point in time when Alice updates her out variable, and fix a corresponding integer $t \leqslant T$ such that $\not\in(R'[1, t])$ is the complete sequence of symbols that Alice has received by that time. Assume that*

$$\|[1, t]\| \geqslant \frac{n+2}{\alpha}, \tag{19}$$

$$\mathrm{corr}[1, t] \leqslant \frac{1}{4} - \alpha. \tag{20}$$

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.

Proof. We will only prove the claim for Alice; the proof of Bob is entirely analogous. Lemma 19 implies the existence of $j' \in E'$ and $j'' \in E''$ such that

$$\overline{f_n} \leqslant j' < t, \tag{21}$$

$$\overline{f_n} \leqslant j'' < t, \tag{22}$$

$$\Delta(S'[j'+1, t], R'[j'+1, t]) < 1 - \alpha, \tag{23}$$

$$\Delta(S''[j''+1, t], R''[j''+1, t]) < 1 - \alpha. \tag{24}$$

By the definition of E' and E'',

$$\mathrm{SD}(S'[1, j'], R'[1, j']) < 1 - \alpha, \tag{25}$$

$$\mathrm{SD}(S''[1, j''], R''[1, j'']) < 1 - \alpha. \tag{26}$$

As a result,

$$\begin{aligned}
&\mathrm{SD}_{\overline{\ast}(S'[1,j'])}(S'[1, t], R'[1, t]) \\
&= \mathrm{SD}_{\overline{\ast}(S'[1,j'])}(S'[1, j']S'[j'+1, t],\ R'[1, j']R'[j'+1, t]) \\
&\leqslant \max\{\mathrm{SD}_{\overline{\ast}(S'[1,j'])}(S'[1, j'], R'[1, j']), \\
&\qquad\qquad \Delta(S'[j'+1, t], R'[j'+1, t])\} \\
&\leqslant \max\{\mathrm{SD}(S'[1, j'], R'[1, j']),\ \Delta(S'[j'+1, t], R'[j'+1, t])\} \\
&< 1 - \alpha,
\end{aligned} \tag{27}$$

where the second step is valid by Proposition 3 (iv), the third step uses (1), and the final step is immediate from (23) and (25).

When Alice updates her *out* variable, the sequence of symbols that she has received is $\not\in(R'[1, t])$. By (27) and Theorem 7, $\textsc{Decode}_{C,\alpha}(\not\in(R'[1, t])) \succeq (\not\in(S'[1, t]))_{\leqslant\overline{\ast}(S'[1,j'])} = \not\in(S'[1, j'])$. Therefore, just prior to updating *out*, Alice is able to correctly recover the prefix $\not\in(S'[1, j'])$ of the sequence of symbols sent to her by Bob. By Lemma 11, this means that she correctly recovers the complete set of edges encoded by the string $\not\in(S'[1, j'])$. By (21), this prefix $\not\in(S'[1, j'])$ 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 $\not\in(S'[1, j'])$ 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 (22), (24), (26), and j'' playing the role of (21), (23), (25), and j', respectively. ∎

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

Proof of Theorem 9. Recall that $n = |\pi|$ denotes the communication cost of the original protocol, and $\epsilon > 0$ is a constant in the statement of Theorem 9. Consider the interactive coding scheme given by Figures 2 and 3, with parameters set according to

$$\alpha = \frac{\epsilon}{4}, \tag{28}$$

$$N = \left\lceil \frac{n+4}{2\alpha} \right\rceil. \tag{29}$$

By (3), 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 (4), 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 (28) and Lemma 17 that

$$\operatorname{corr}[1, T] \leqslant \frac{1}{4} - 2\alpha. \tag{30}$$

By (29) and Lemma 14,

$$|\!|\!|[1, T]|\!|\!| > \frac{n+4}{\alpha} \tag{31}$$

and therefore

$$|\!|\!|[1, T-1]|\!|\!| > \frac{n+2}{\alpha}. \tag{32}$$

Also,

$$\begin{aligned} \operatorname{corr}[1, T-1] &\leqslant \frac{|\!|\!|[1, T]|\!|\!|}{|\!|\!|[1, T-1]|\!|\!|} \cdot \operatorname{corr}[1, T] \\ &\leqslant \left(1 + \frac{2}{|\!|\!|[1, T-1]|\!|\!|}\right) \cdot \operatorname{corr}[1, T] \\ &\leqslant \left(1 + \frac{2\alpha}{n+2}\right) \cdot \left(\frac{1}{4} - 2\alpha\right) \\ &\leqslant \frac{1}{4} - \alpha, \end{aligned} \tag{33}$$

where the third step uses (30) and (32). 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 $\not\in(R'[1, T-1])$ or $\not\in(R'[1, T])$. Likewise, the complete sequence of symbols that Bob has received at the time of his last update is $\not\in(R''[1, T-1])$ or $\not\in(R''[1, T])$. By (30)–(33) and Theorem 20, 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 9. ∎

IV. A coding scheme with a constant-size alphabet

In the full version of this paper, we adapt the proof of Theorem 9 to use an alphabet of constant size. This modification yields the main result of our work, which we restate here for the reader's convenience.

Theorem 21. *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*

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

At a high level, our proof of Theorem 21 is similar to the proof of Theorem 9 in the previous section, and we are 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.

Acknowledgments

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. The authors are thankful to Mark Braverman, Rafail Ostrovsky, and the anonymous reviewers of FOCS 2017 for their valuable feedback on an earlier version of this manuscript.

References

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[2] ——, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, October 1948.

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

[4] L. J. Schulman, "Communication on noisy channels: A coding theorem for computation," in *FOCS*, 1992, pp. 724–733.

[5] ——, "Deterministic coding for interactive communication," in *STOC*, 1993, pp. 747–756.

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

[7] R. Gelles, "Coding for interactive communication: A survey," 2015, available at http://www.eng.biu.ac.il/~gellesr/survey.pdf.

[8] 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.

[9] M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky, "Coding for interactive communication correcting insertions and deletions," in *ICALP*, 2016, pp. 61:1–61:14.

[10] 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.

[11] A. A. Sherstov and P. Wu, "Optimal interactive coding for insertions, deletions, and substitutions," in *Electronic Colloquium on Computational Complexity (ECCC)*, May 2017, report TR17-079.

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

[13] S. Agrawal, R. Gelles, and A. Sahai, "Adaptive protocols for interactive communication," in *ISIT*, 2016, pp. 595–599.