Interactive Coding Resilient to an Unknown Number of Erasures

Ran Gelles
Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel
ran.gelles@biu.ac.il

Siddharth Iyer
University of Washington, WA, USA[1]
siyer@cs.washington.edu

Abstract

We consider distributed computations between two parties carried out over a noisy channel that may erase messages. Following a noise model proposed by Dani et al. (2018), the noise level observed by the parties during the computation in our setting is arbitrary and a priori unknown to the parties.

We develop *interactive coding schemes* that adapt to the actual level of noise and correctly execute any two-party computation. Namely, in case the channel erases T transmissions, the coding scheme will take $N + 2T$ transmissions using an alphabet of size 4 (alternatively, using $2N + 4T$ transmissions over a binary channel) to correctly simulate any binary protocol that takes N transmissions assuming a noiseless channel. We can further reduce the communication to $N + T$ by relaxing the communication model and allowing parties to remain silent rather than forcing them to communicate in every round of the coding scheme.

Our coding schemes are efficient, deterministic, have linear overhead both in their communication and round complexity, and succeed (with probability 1) regardless of the number of erasures T.

2012 ACM Subject Classification Theory of computation → Interactive computation; Mathematics of computing → Coding theory; Computing methodologies → Distributed algorithms

Keywords and phrases Interactive coding, erasure channels, distributed computation with noise, unbounded noise

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2019.13

Related Version A preliminary full version of this work [18] is available at `https://arxiv.org/abs/1811.02527`.

Funding *Ran Gelles*: Research supported in part by the Israel Science Foundation (ISF) through grant No. 1078/17.

Acknowledgements We thank Amir Leshem for plenty of helpful discussions.

1 Introduction

Consider two remote parties that use a communication channel in order to perform some distributed computation. One main obstacle they may face is noise added by the communication channel, corrupting their messages and ruining the computation. In the early 90's, Schulman [29, 30] initiated the field of *interactive coding* where the parties use coding techniques in order to complete their computation correctly despite possible communication noise.

Channels may introduce different types of noise. Among the common noise types are *substitution* noise, where the channel changes the content of messages (e.g., it flips communicated bits), *insertions and deletions*, where the channel may introduce a new message

[1] Part of this work was done while at Bar-Ilan University and a student at Birla Institute of Technology and Science, Pilani, India.

23rd International Conference on Principles of Distributed Systems (OPODIS 2019).
Editors: Pascal Felber, Roy Friedman, Seth Gilbert, and Avery Miller; Article No. 13; pp. 13:1–13:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

or completely remove a transmission, and *erasures* where the channel erases transmissions, i.e., replacing them with an erasure mark. Throughout the last several years many interactive coding schemes were developed, allowing parties to perform computations over the various channels and noise types, e.g., [9, 12, 32], see related work below and [17] for a survey.

Naturally, some bounds on the noise must be given. As a trivial example, it is clear that if *all* the transmissions are corrupted, there is no hope to complete any distributed computation. Braverman and Rao [9] showed that for substitution noise, a noise fraction of $1/4$ is maximal. That is, as long as the noise is limited to corrupt less than $1/4$ of the communication, there are coding schemes that will succeed in producing the correct output. However, if the noise exceeds this level, *any* coding scheme for a general computation is bound to fail. Similarly, noise rate of $1/4$ is maximal (and achievable) for insertion and deletion noise [7, 32], and noise rate of $1/2$ is maximal (and achievable) for erasure noise [14, 12].

All the above schemes must know *in advance* the amount of noise they are required to withstand. For instance, the scheme of [9] is given some parameter $\rho < 1/4$, which stands for the (maximal) fraction of noise that instance will be able to handle. Given this parameter and the length of the (noiseless) computation to be performed N, the coding scheme determines how many transmissions it should take in order to perform the computation, say, $\tilde{N} = \tilde{N}(\rho, N)$ transmissions. It is then guaranteed that as long as the noise corrupts at most $\rho\tilde{N}$ transmissions, the computation succeeds.

In a recent work, Dani et al. [11] considered the case where the noise amount may be arbitrary and *a priori* unknown to the coding scheme. That is, the channel may corrupt up to T transmissions, where $T \in \mathbb{N}$ is some fixed amount of noise which is independent of the other parameters of the scheme and the computation to be performed. The work of Dani et al. [11] considered substitution noise and showed a scheme that succeeds with high probability and, if the channel corrupts T transmissions during the execution of the scheme, then the scheme will take $N + O(T + \sqrt{TN \log N})$ transmissions to conclude.

Again, some limitations must be placed on the noise. Indeed, assume that an uncorrupted computation on the input (x, y) terminates in $\tilde{N}$ rounds, and assume $T > \tilde{N}$. Then, a substitution noise can always make Alice (wlog) believe that Bob holds y, by corrupting the messages Alice receives in the first $\tilde{N}$ rounds. Note that Alice then terminates with the wrong output, i.e., the output for (x, y). Dani et al. [11] dealt with the above impossibility by assuming that the parties have access to some shared randomness and that the adversary is *oblivious* to that randomness. This allowed them to employ cryptographic tools in order to guarantee the authentication of communicated messages.

In this work we focus on *erasure noise*, where the channel either transmits the message as is, or erases the message and outputs a special $\perp$ symbol that indicates this event. Erasure noise naturally appears in many practical situations, e.g., when an Ethernet packet gets corrupted yet this corruption is detected by the CRC checksum mechanism [25]. In this case the packet is considered invalid and will be dropped. This situation is equivalent to an erasure of that transmission. We note that this type of noise is weaker than substitution noise. This allows us to obtain coding schemes without further assumptions, mainly, without the need for shared randomness and without requiring the adversary to be oblivious.

Our main result is an efficient and deterministic coding scheme that withstands an arbitrary and *a priori* unknown T erasures with probability 1.

▶ **Theorem 1** (main, informal). *Given any two-party binary interactive protocol π of length N, there exists an efficient and deterministic noise-resilient protocol Π_4 of length $N + 2T$ that simulates π over a 4-ary channel in the presence of an arbitrary and a priori unknown number T of erasures.*

Alternatively, there exists an efficient, deterministic, binary *noise-resilient protocol* Π_2 *of length* $2N + 4T$ *that simulates* π *assuming an arbitrary and a priori unknown number* T *of erasures. It holds that*

$$\mathsf{CC}(\Pi_4) = \mathsf{CC}(\Pi_2) = 2N + 4T.$$

Since T, the amount of noise, is unknown to begin with, the length of the coding scheme must adapt to the actual noise that the parties observe. Such coding schemes are called *adaptive* [3]. Adaptivity raises several issues that must be dealt with appropriately. The main issue is *termination.* Since the coding scheme adapts its length to the observed noise, and since the different parties observe different noise patterns, their termination is not necessarily synchronized. As a matter of fact, obtaining synchronized termination is impossible.

▶ **Lemma 2.** *Let* Π *be a protocol for exchanging messages between Alice and Bob, that is resilient to an unbounded amount* T *of erasures. Then, there always exists a noise pattern for which Alice and Bob terminate in different rounds.*

This can be seen as a variant of the famous "coordinated attack problem" [24], where reaching full synchronization between two parties is known to be impossible. See [18] for a proof and an elaborated discussion about unsynchronized termination.

Unsynchronized termination means that one party may terminate while the other party continues to send (and receive) transmissions as dictated by its protocol. In this case, the communication model should specify what happens in those rounds where only one party is active and the other has terminated. Specifically, it should specify what messages the active party receives in this case.

In our setting we define a special symbol we call *silence* (cf. [3]). We assume that silence is (implicitly) communicated by a terminated party. That is, the still-active party hears silence in every round it is set to listen and the other party has already terminated. We note that silence is corruptible – the channel may erase silent transmissions, and the active party will see an erasure mark instead. On the other hand, these implicit silent transmissions are not considered part of the communication of the protocol (i.e., we do not count them towards the communication complexity). Hearing a silence is a univocal indication that the other side has terminated, allowing the other party to terminate as well and bypassing the impossibility of synchronized termination.

In the setting of Theorem 1 we do not allow the parties to remain silent before termination – in each timestep a party is set to speak it must communicate a valid message. However, in today's networks, especially in networks with multiple parties, it is very common that parties send messages only if they have information to send, and keep silent otherwise.

Our second result extends Theorem 1 to the setting where parties are allowed to either speak or remain silent at every timestep (called the AGS setting hereinafter, see [3]). In this case, termination becomes even more tricky. Recall that the coding scheme of Theorem 1 uses silence as an indicator for termination. We can do the same in the AGS setting and avoid using silence throughout the protocol, keeping it as an indicator towards termination only. However, this would effectively reduce the AGS setting to the one of Theorem 1, and lead to a suboptimal scheme.

Instead, we take a different approach that requires parties to remain silent during the protocol in certain cases. This has the effect of reducing the communication at the expense of not being able to identify termination at times. In particular, one of the parties may remain active indefinitely. However, that party will remain silent after the other party has terminated, and moreover, it will hold the correct output of the computation. We call this situation *semi-termination.*

▶ **Definition 3.** *We say that a protocol has semi-termination if there exists a round t after which both the following conditions hold:*

1. *Both parties have computed the correct output, and*
2. *Both parties remain silent indefinitely (whether they terminate or not)*

Our result for this setting is as follows.

▶ **Theorem 4** (AGS setting, informal)**.** *Given any binary two-party interactive protocol π of length N, there exists an efficient and deterministic protocol Π_4 in the AGS setting with semi-termination, of length $N + 4T$ sending at most $N + T$ symbols from alphabet of size 4, that simulates π assuming an arbitrary and a priori unknown number T of erasures.*

Alternatively, there exists an efficient and deterministic protocol Π_1 in the AGS setting with semi-termination, with length $4N + 16T$ that communicates at most $N + T$ unary (i.e., non-silent) symbols and simulates π assuming an arbitrary and a priori unknown number T of erasures.

In this setting, it is significant to bound the round complexity since silence is not counted towards the communication complexity, yet it can be used to transfer information. As stated in the above theorem, the round complexity of the resilient protocols Π_4 and Π_1 until the event of semi-termination, is linear in the round complexity of the noiseless protocol π, namely, $\mathsf{RC}(\Pi_4) = O(N + T)$ and $\mathsf{RC}(\Pi_1) = O(N + T)$.

Organization. In the next part we overview the techniques used to obtain Theorem 1 and Theorem 4. Section 2 formally defines the communication and noise model, and fixes some notations used throughout. In Section 3 we describe the noise resilient protocol of Theorem 1 and analyze its correctness. The coding scheme of Theorem 4 along with its correctness is deferred to the full version of this paper (see [18] for a preliminary version). Further, in [18] we discuss the issue of unsynchronized termination and prove that it is impossible to obtain synchronized termination when the noise is unbounded.

1.1 Coding Schemes Overview

Erasure noise has two attractive properties we utilize towards our scheme. The first is that, if there was a corruption, the *receiver* is aware of this event; the second property is that, if there was no corruption, the received message is the *correct* one. Our scheme follows a technique by Efremenko et al. [12], where the parties simulate the noiseless protocol π bit by bit. As long as there is no noise, they can carry out the computation identically to π. However, in the case of an erasure, the receiver needs to signal the other side that it did not receive the last message and request a retransmission. The main problem is that this request message may get erased as well, making both sides confused regarding to what should be sent next.

In [12] this issue is solved by extending each message by two bits that indicate the round currently being simulated. It is proven that the parties may simulate different rounds, however, the discrepancy in the round number is bounded by 1. Hence, a round number modulus 3 is required in [12] to indicate whether Alice is ahead, Bob is ahead, or they are at the same round.

Our scheme combines the above technique with a challenge-response technique employed by Dani et al. in [11], in order to obtain resilience against an unbounded number of erasures. Our coding scheme is *not* symmetric, but rather Alice always begins a round by sending a "challenge" message, followed by Bob replying with a response. Alice then determines whether the challenge-response round was successful: if both messages were not erased, Alice would see the correct response from Bob and would deduce that both messages were received correctly. If Alice received an erasure, or if she received the wrong response, she would deduce that an erasure has occurred during this round and the round should be re-simulated.

Bob, similarly but not identically, gets the challenge message from Alice and verifies that it belongs to a new round (that is, the challenge differs from the previous round). If this is the case, he replies with the next bit. Otherwise, i.e., if Alice's challenge was erased, or if Bob receives the challenge of the previous round, he replies with the response of the *previous* round. Note that if Alice did send a new challenge and it was erased, she will now get the response of the previous round and realize there is a mismatch.

It is not too difficult to see that the "challenge" suffices to be a single bit – the current simulated round number in π, modulus 2 (which we call the *parity* from now on). Since the scheme is not symmetric, it can never happen that Alice has advanced to the next round while Bob has not. The other case, where Bob is ahead of Alice by a single round is still possible. Therefore, one bit of information suffices to distinguish between these two cases.

In more details, Alice begins a round by sending Bob the next bit of the simulation of π, along with the parity of the round number (in π) of that bit. If Bob receives this message and the parity matches the round number he expects, he records the bit sent by Alice and replies with the next bit of π using the same parity. If this message reaches Alice correctly, she knows the round is over and advances to the next round. If Bob does not receive Alice's message (i.e., it gets erased), or if the parity is incorrect, Bob replies with the bit of the previous round along with the parity of that round. Similarly, if Alice receives a message with a wrong parity or an erased message, she keeps re-simulating the same round, until she gets the proper reply from Bob.

Note that a single erasure delays the progress of the simulation by a single round (2 transmissions). However, once there is a round in which both messages are not erased, the simulation correctly continues, and the succeeding two bits of π are correctly simulated. That is, as long as there is noise, the simulation just hangs, and when the noise ceases, the simulation continues from exactly the same place it stopped.

Once Alice completed simulating the last round of π, she quits. Recall that Bob is always ahead of Alice, thus if Alice completes the simulation, so does Bob. After Alice terminates, Bob receives silence unless erased by the channel. When Bob hears a silence, he learns that Alice has terminated and quits the scheme as well. The noise may delay Bob's time of termination by corrupting the silence, however, once the noise is over, Bob will learn that the simulation has completed and will quit as well.

Coding schemes for parties who may keep silent

Our second scheme works in the communication model where parties are allowed to remain silent if they wish (the AGS setting). The main advantage in being able to remain silent is allowing the parties to communicate information in an optimized manner which reduces their communication complexity. Specifically, consider the above idea of challenge-response, where a party replies with the wrong response in order to indicate there was an error and that a round should be re-simulated. In the AGS setting we can instead keep silent in order to signal this retransmission request.

The idea is as follows. Similar to the scheme above Alice begins a round by sending her bit to Bob (along with the parity). If the transmission is received correctly, Bob replies with his next bit. If Bob receives an erasure instead, he remains silent. This signals Alice that an error has happened and that she should re-simulate the last round. Similarly, if Alice sees an erasure she keeps silent. In all other cases, i.e., receiving a silence or receiving a wrong parity, the parties re-transmit their last message as before.

The effect of a party keeping silent for asking a retransmission is reducing the communication complexity. Note that each erasure causes the recipient to remain silent for one single round, instead of sending a message that indicates an erasure. Then, the simulation continues from the point it stopped. On the other hand, the analysis becomes slightly more difficult in this case since silence may be erased as well, causing the other side to remain silent and signal there was an erasure. This may cause the first party to repeat its message while the other side should have actually resent its message in order to advance the simulation. Luckily, this issue does not falsify the correctness of the simulation – as a result of sending the parity, the extra retransmission is simply ignored. Furthermore, such a superfluous transmission does not increase the communication overhead since it can only happen when multiple erasures have occurred in the same round or in consecutive rounds.

When silence has a meaning of requesting a retransmission, we cannot use it anymore to indicate termination. Note that whatever Alice sends Bob to inform him she is going to terminate may get erased, so if Alice terminates Bob will not be aware of this fact and will remain active. If Alice waits to hear a confirmation from Bob that he received the indication and learned that Alice is about to terminate – this confirmation may get erased. Bob never learns if Alice has received his acknowledgment or not; then, if Bob assumes that Alice has terminated and terminates himself, it will be Alice who hangs in the protocol waiting for Bob's confirmation, and so on. Our approach to this conundrum is to allow the parties not to terminate as long as there exists a point in time beyond which the parties remain silent and both hold the correct output. In our scheme, Alice will actually terminate once she learns that the simulation is done[2] (recall that if Alice completed the simulation then Bob completed it as well, but the other direction does not necessarily hold). Bob's actions in the final part of the protocol are slightly different from his normal behaviour. Once Bob completed simulating π, but he is unaware whether or not Alice completed simulating π, he keeps silent unless he hears a message from Alice that re-simulates the final round. In this case, he replies with his final bit. In all other cases he remains silent.

1.2 Related Work

The field of interactive coding was initiated by the seminal work of Schulman [29, 30] focusing on two parties that communicate over a binary channel with either substitution noise (random or adversarial) or with erasure noise. Followup work (for substitution noise) developed coding schemes with optimal resilience [9, 6, 20] efficiency [31, 8, 19, 4, 20, 16], or good rate [26, 22, 16]. Coding schemes for different channels and noise types were developed in [28, 12, 15] for channels with feedback, in [7, 32, 23, 13] for insertions and deletions noise, and in [5, 27] for quantum channels. Interactive coding over channels that introduce erasure noise was explored in [30, 28, 14, 15, 12, 3]. In particular, Efremenko et al. [12] developed efficient coding schemes for optimal erasure rates, and Gelles and Haeupler [15] developed efficient coding schemes with optimal rate assuming a small fraction of erasures. Adaptive models were considered in [3, 21, 20]. See [17] for a survey on the field of interactive coding.

[2] It is possible to let Alice send Bob a "termination message" right before she quits in order to signal Bob she is about to terminate. In case this message is not erased, Bob will terminate as soon as he hears this special message. Otherwise, he will keep running the scheme but remain silent. In either case, the protocol satisfies the semi-termination requirements.

Closest to our work is the work of Dani et al. [11] who considered the case of an arbitrary noise amount that is unknown to the scheme. Their coding scheme assumes substitution noise which is oblivious to the randomness used by the parties as well as to the bits communicated through the channel. An AMD code [10] is used to fingerprint each transmitted message, allowing the other side to detect corruptions with high probability. Aggrawal et al. [2] use similar techniques to develop a robust protocol for message transfer, assuming bi-directional channel that suffers from an arbitrary (yet finite) and unknown amount of bit flips. Aggarwal et al. [1] extended this setting to the multiparty case, where n parties, rather than two, perform a computation over a noisy network with an arbitrary and unknown amount of noise.

2 Model Definition

Standard notations. For an integer $i \in \mathbb{N}$ we denote by $[i]$ the set $\{1, 2, \ldots, i\}$. All logarithms are taken to base 2. The concatenation of two strings x and y is denoted $x \circ y$. We let λ denote the empty string.

Interactive computation. In our setting, two parties, Alice and Bob, possess private inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$, respectively, and wish to compute some predefined function $f(x, y)$. The computation is performed by exchanging messages over a channel with fixed alphabet Σ. The computation is specified by a synchronous *interactive protocol* π. An interactive protocol $\pi = (\pi_A, \pi_B)$ is a pair of algorithms that share a common clock, and specify, for each timestep and each one of the parties, the following details: (1) which party speaks and which party listens in this time-step; (2) if the party is set to speak, which symbol to communicate; (3) whether the party terminates in this timestep, and if so, what the output is.

Without loss of generality, we assume that the (noiseless) protocol π is alternating, i.e., Alice and Bob speak in an alternating manner, Alice speaks in odd rounds and Bob in even rounds. If this is not the case, it can be made so while increasing the communication by a factor of at most 2. We define a *round* of the noiseless protocol to be a sequence of two time steps in which two consecutive messages are sent: the first is sent by Alice and the second by Bob. For example, the first round consists of the first message sent by Alice and the subsequent message from Bob. In general for $r \geq 1$, after $r - 1$ rounds have elapsed, the r^{th} round consists of the r^{th} message sent by Alice and Bob's subsequent message. For the sake of convenience, we assume that the last message of the protocol is sent by Bob; this can be ensured by padding the protocol by at most one bit. Since the protocol is alternating we can also think of rounds being associated with timesteps. More formally, in the r^{th} round, Alice and Bob send messages corresponding to the $(2r-1)^{th}$ and the $2r^{th}$ timestep respectively. We say that a protocol has length N if Alice and Bob exchange N messages in the protocol.

Given a specific input (x, y) the transcript $\pi(x, y)$ is the concatenation of all the messages received during the execution of π on (x, y).

Erasures. The communication channel connecting Alice and Bob is subject to *erasure* noise. In each timestep, the channel accepts a symbol $s \in \Sigma$ and outputs either s or a special erasure mark ($\perp \notin \Sigma$). The noise is assumed to be worst-case (adversarial), where up to T symbols may be replaced with erasure marks. The value of T is unbounded and *unknown* to the parties – their protocol should be resilient to any possible $T \in \mathbb{N}$. A *noise pattern* is a bit-string $E \in \{0,1\}^*$ that indicates erasures in a given execution instance of the protocol. Specifically, there is an erasure in the i-th timestep if and only if $E_i = 1$. Given a specific instance where both parties terminate before its s-th transmission, the number of erasures that are induced by E on that instance is the Hamming weight of $E_1, \ldots, E_s$, i.e., the number of 1's in the bit-string. We sometime allow E to be of infinite length, however our protocols will be resilient only against noise patterns with bounded amount of noise (i.e., if E is infinite, then its suffix is required to be the all-zero string; we call such noise *finite* noise).

Coding scheme: Order of speaking, silence and termination. A *coding scheme* is a protocol Π that takes as an input another protocol π (that assumes noiseless channels) and simulates π over a noisy channel. By saying that Π *simulates* π over an erasure channel with (unbounded) T corruptions, we mean the following. For any pair of inputs (x, y) given to Alice and Bob, after executing Π in the presence of T erasures, Alice and Bob will produce the transcript $\pi(x, y)$. The above should hold for *any* $T \in \mathbb{N}$, independent of π, x, y and unknown to Π (i.e., to the parties).

We assume that the protocol Π, at any given timestep, exactly one party can be the sender and the other party is the receiver. That is, it is never the case that both parties send a symbol or both listen during the same timestep. On the other hand we do not assume that the parties terminate together, and it is possible that one party terminates while the other does not. In this case, whenever the party that hasn't yet terminated is set to listen, it hears some default symbol $\square$, which we call *silence*.

There are several ways to treat silence. One option, taken in [3], is to treat the silence similar to any other symbol of Σ. That is, as long a party has not terminated and is set to speak, it can either send a symbol or remain silent ("send $\square$") at that round.[3] A different approach would be to require the parties to speak a valid symbol from Σ while they haven't terminated. That is, a party cannot remain silent if it is set to speak; this prevents the parties from using silence as a means of communicating information during the protocol. Once a party terminates, and only then, silence is being heard by the other party. This mechanism makes it easier for the parties to coordinate their termination. In particular, the event of termination of one party transfers (limited) information to the other party. We take this approach in the scheme of Section 3.

Another subtlety stems from the fact that the length of the protocol is not predetermined. That is, the length of the protocol depends on the actual noise in the specific instance. Such protocols are called adaptive [3]. In this case it makes sense to measure properties of the protocol with respect to a specific instance. For instance, given a specific instance of the protocol Π on inputs (x, y) with some given noise pattern, the communication complexity $\mathsf{CC}(\Pi(x, y))$ is the number of symbols *sent* by both parties in the specific instance. The communication is usually measured in bits by multiplying the number of symbols by $\log |\Sigma|$. The noise in a given instance is defined to be the number of corrupted transmissions until both parties have terminated, *including* corruptions that occur after one party has terminated and the other party has not. Corruptions made after both parties have terminated cannot affect the protocol, and we can assume such corruptions never happen.

3 A coding scheme for an unbounded number of erasures

In this section we provide a coding scheme that takes as an input any noiseless protocol π and simulates it over a channel that suffers from an unbounded and unknown number of erasures T. The coding scheme uses an alphabet Σ of size 4; in this setting parties are not allowed to be silent (unless they quit the protocol) and in every round they must send a symbol from Σ.

[3] We take this approach in the coding scheme designed to prove Theorem 4, see [18].

3.1 The Coding Scheme

The coding scheme for Alice and Bob, respectively, is depicted in Algorithms 1 and 2.

Inspired by the simulation technique of [12], our simulation basically follows the behavior of the noiseless protocol π step by step, where Alice and Bob speak in alternating timesteps. In each timestep, the sending party tries to extend the simulated transcript by a single bit. To this end, the parties maintain partial transcripts $\mathcal{T}^A$ for Alice (and $\mathcal{T}^B$ for Bob) which is the concatenation of the information bits of π that the parties have simulated so far and are certain of. Then, if Alice is to send a message to Bob, she generates the next bit of π given her partial simulated transcript, i.e., $\pi(x \mid \mathcal{T}^A)$, and sends this information to Bob.

In addition to the information bit, Alice also sends a parity of the round number she is currently simulating. That is, Alice holds a variable r_A which indicates the round number she is simulating. Recall that each round contains two timesteps, where Alice communicates in the first timestep and Bob in the second. Alice sends Bob the next bit according to her $\mathcal{T}^A$ and waits for Bob's reply to see if this round was successfully simulated. If Bob's reply indicates the same parity (i.e., the same round), Alice knows her message arrived to Bob correctly and hence the round was correctly simulated. In this case Alice increases r_A. Otherwise, she assumes there was a corruption and she keeps r_A as is; this causes the same round of π to be re-simulated in the next round of the simulation protocol.

Bob holds a variable r_B which again holds the (parity of) the latest round in π he has simulated. In a somewhat symmetric manner (but not identical to Alice!), he expects to receive from Alice the bit of the *next* round of π, $r_B + 1$. If this is the case he responds with his bit of that same round, or otherwise he re-transmits his bit of round r_B.

Our coding scheme assumes a channel with alphabet $\Sigma = \{0,1\} \times \{0,1\}$, where every non-silent message can be interpreted as $m = (\mathsf{Info}, \mathsf{Parity})$, where $\mathsf{Info} \in \{0,1\}$ is the information bit (simulating π) and $\mathsf{Parity} \in \{0,1\}$ is the parity of the round of π simulated by the sender.

The above continues until Alice has simulated the entire transcript of π, i.e, when r_A reaches the number of rounds in π. At this point, Alice exits the protocol. Bob, however, cannot tell whether Alice has completed the simulation or not and waits until he sees a silence, which indicates that Alice has terminated, only then does he exit the protocol. As regards the correctness, we prove that in any round of the simulation, Bob has seen at least as much of the noiseless protocol that Alice herself has seen. In particular, when Alice exits, Bob must have seen the entire transcript of the noiseless protocol.

3.2 Analysis

Preliminaries. Recall that in our terminology a round consists of two timesteps, where at every timestep one party sends one symbol from Σ. Alice sends symbols in odd timesteps and Bob in even ones. The above applies for both the noiseless protocol π (where the alphabet is binary) and for the coding scheme Π given by Algorithms 1 and 2 (where $\Sigma = \{0,1\} \times \{0,1\}$).

We think of the communication transcript as a string obtained by the concatenation of symbols sent during the course of the protocol run. Given a noiseless protocol π and inputs x, y, we denote by $m_A^\pi(i)$ (respectively, $m_B^\pi(i)$) the message sent by Alice (respectively, Bob) in the i-th round in the noiseless protocol π. Let $\mathcal{T}^\pi(r)$ be the transcript of the players after r rounds in π. The length of π is denoted N and without loss of generality we will assume that N is even.

We start our analysis by fixing a run of the coding scheme Π, specified by fixing an erasure pattern and inputs (x, y). Let k be the number of timesteps in the given run, and note that k is always odd. We define rounds in the coding scheme Π in the same way as we

Algorithm 1 Simulation over Erasure Channel with Unbounded Noise (Alice).

```
     Data: An alternating binary protocol $\pi$ of length $N$ and an input $x$.
A.1  Initialize $\mathcal{T}^A \leftarrow \lambda$, $r_A \leftarrow 0$.
A.2  while ($r_A < \frac{N}{2}$) do
         // Send a Message (odd time-step)
A.3      $r_A \leftarrow r_A + 1$
A.4      $b_{send} \leftarrow \pi(x \mid \mathcal{T}^A)$
A.5      $\mathcal{T}^A \leftarrow \mathcal{T}^A \circ b_{send}$
A.6      send $(b_{send}, r_A \mod 2)$
         // Receive a Message (even time-step)
A.7      Obtain $m' = (b_{rec}, r_{rec})$
A.8      if $m' \neq \perp$ and $r_{rec} = r_A \mod 2$ then
A.9          $\mathcal{T}^A \leftarrow \mathcal{T}^A \circ b_{rec}$
         else
A.11         Delete last symbol of $\mathcal{T}^A$
A.12         $r_A \leftarrow r_A - 1$
         end
     end
A.13 Output $\mathcal{T}^A$
```

define rounds in the noiseless protocol. That is, for $i \in \{1, 2, \ldots, \lceil \frac{k}{2} \rceil\}$, round i (in the coding scheme) corresponds to the timesteps $2i - 1$ and $2i$. Throughout this section by round i we refer to the round in the coding scheme unless it is specified otherwise. For the sake of clarity, we maintain that round i begins at the start of the $(2i-1)$-th timestep and ends at the ending of the $2i$-th timestep. Since Π is alternating, Alice begins a round by sending a message and the round ends after Alice receives Bob's response.

We define t_A (and t_B) to be the termination round of Alice (and respectively, Bob). Observe that $t_A < t_B = \lceil \frac{k}{2} \rceil$ since Bob terminates only once he hears silence, which can happen only if Alice has already terminated in a previous round. For any round i and variable v we denote by $v(i)$ the value of v at the beginning of round i. In particular, $r_A(i)$ and $r_B(i)$ are the values of r_A and r_B at the beginning of round i. Since r_A is not defined after Alice exits in Algorithm 1, for all $t_A < i \leq t_B$, we set $r_A(i)$ to be the value of r_A at the *end* of round t_A, that is, its value just before Alice quits. We similarly define $\mathcal{T}^A(i)$ for $t_A < i \leq t_B$. Finally, let $m_A(i) = (b_A(i), \rho_A(i))$ and $m_B(i) = (b_B(i), \rho_B(i))$ be the messages sent by Alice and Bob (see Lines A.6 or B.13 or B.15) respectively in round i.

Technical lemmas and proof of correctness. The main technical claim of this part is Lemma 7, which we will now prove. We begin with the simple observation that, in every round, the parties' transcripts (and the respective round number the parties believe they simulate) either increase by exactly the messages exchanged during the last round, or they remain unchanged.

▶ **Lemma 5.** *For any $i \in [t_B]$, the following holds.*

1. $r_A(i+1) \in \{r_A(i), r_A(i)+1\}$ *and* $\mathcal{T}^A(i+1) \in \left\{\mathcal{T}^A(i), \mathcal{T}^A(i) \circ b_A(i) \circ b_B(i)\right\}$ *and,*
2. $r_B(i+1) \in \{r_B(i), r_B(i)+1\}$ *and* $\mathcal{T}^B(i+1) \in \left\{\mathcal{T}^B(i), \mathcal{T}^B(i) \circ b_A(i) \circ b_B(i)\right\}$.

Furthermore, r_A (r_B) changes if and only if $\mathcal{T}^A$ ($\mathcal{T}^B$) changes.

Algorithm 2 Simulation over Erasure Channel with Unbounded Noise (Bob).

```
      Data: An alternating binary protocol $\pi$ of length $N$ and an input $y$.
B.1   Initialize $\mathcal{T}^B \leftarrow \lambda$, $r_B \leftarrow 0$, $err \leftarrow 0$, $m \leftarrow (0,0)$
B.2   while $m' \neq \Box$ do                              // while Silence isn't heard
          // Receive a Message (odd time-step)
B.3       Obtain $m' = (b_{rec}, r_{rec})$
B.4       if $m' = \bot$ or $r = r_B \mod 2$ then
B.5           $err \leftarrow 1$                           // error detected
B.6       else
B.7           $\mathcal{T}^B \leftarrow \mathcal{T}^B \circ b_{rec}$
B.8           $err \leftarrow 0$
          end
          // Send a Message (even time-step)
B.9       if $err = 0$ then
B.10          $b_{send} \leftarrow \pi(y \mid \mathcal{T}^B)$
B.11          $\mathcal{T}^B \leftarrow \mathcal{T}^B \circ b_{send}$
B.12          $r_B \leftarrow r_B + 1$
B.13          $m \leftarrow (b_{send}, r_B \bmod 2)$
B.14      else                                     // $err = 1$: keep $r_B$ and $m$ unchanged.
B.15          send $m$
          end
      end
B.16  Output $\mathcal{T}^B$
```

Proof. Consider Algorithm 2. It is immediate that r_B can increase by at most 1 in every round. In round i Bob starts by either appending $b_{rec} = b_A(i)$ to $\mathcal{T}^B(i)$ and setting $err = 0$; or keeping $\mathcal{T}^B(i)$ unchanged and setting $err = 1$. In the former case, Bob appends $b_{send} = b_B(i)$ to $\mathcal{T}^B(i) \circ b_A(i)$ and increases r_B. Otherwise, he keeps $\mathcal{T}^B(i)$ unchanged, and r_B remains the same as well.

In Algorithm 1 Alice may decrease her r_A, but note that she always begins round $i \leq t_A$ by increasing it (Line A.3) and appending $b_{send} = b_A(i)$ to $\mathcal{T}^A(i)$ (Line A.5). She then either decreases r_A back to what it was and in this case she erases $b_A(i)$ (see Lines A.12 and A.11), or she keeps it incremented and appends $b_{rec} = b_B(i)$ to $\mathcal{T}^A(i) \circ b_A(i)$. For $i > t_A$, i.e., after Alice terminates, the above trivially holds. ◀

▶ **Corollary 6.** *For any two rounds $i \leq j$, $\mathcal{T}^A(i) = \mathcal{T}^A(j)$ if and only if $r_A(i) = r_A(j)$. Similarly, $\mathcal{T}^B(i) = \mathcal{T}^B(j)$ if and only if $r_B(i) = r_B(j)$.*

Proof. Note that $\mathcal{T}^A, r_A$ are non-decreasing. Lemma 5 proves that these two variable increase simultaneously, which proves this corollary. The same holds for $\mathcal{T}^B, r_B$. ◀

▶ **Lemma 7.** *For $i \in [t_A + 1]$, one of the following conditions holds:*

1. *$r_B(i) = r_A(i)$ and $\mathcal{T}^B(i) = \mathcal{T}^A(i)$ or,*
2. *$r_B(i) = r_A(i) + 1$ and $\mathcal{T}^B(i) = \mathcal{T}^A(i) \circ b_A(i-1) \circ b_B(i-1)$.*

Furthermore, in either case, $\mathcal{T}^A$ and $\mathcal{T}^B$ are prefixes of $\mathcal{T}^\pi$.

Proof. We prove the lemma by induction on the round i. In the first round, $r_A(1) = r_B(1) = 0$ and $\mathcal{T}^B(1) = \mathcal{T}^A(1) = \lambda$ (see Lines A.1, B.1), thus Item 1 is satisfied. Note that $\mathcal{T}^A$ and $\mathcal{T}^B$ are trivially the prefixes of $\mathcal{T}^\pi$. Now assume that the conditions hold at (the beginning of) round $i \leq t_A$ and consider what happens during this round. Note that both parties are active during round i.

Case 1: $r_A(i) = r_B(i)$**.** Suppose Alice receives an (uncorrupted) message from Bob that carries the parity $r_A(i)+1$ (i.e., when Alice executes Line A.8 and not the else part of Lines A.11–A.12). This means that Bob sends a message with parity $r_A(i)+1 = r_B(i)+1$. Since $r_B(i) = r_A(i)$, this can only happen if Bob executed Lines B.13 and B.12, hence, $r_B(i+1) = r_B(i)+1$. If Alice receives a corrupted message, she decrements r_A (that she had increased at the beginning of the round) and also deletes the last message from her transcript. Bob however may have received Alice's message correctly and in that event, he will increment his value of r_B and update his transcript.

Note that if $r_B(i+1) = r_A(i+1)$, then by the induction hypothesis and Lemma 5 both transcripts either remained the same in round i (so they are still the same at the beginning of round $i+1$), or both transcripts increased by appending $b_A(i) \circ b_B(i)$ to each, so they are still equal. Here, $b_A(i) = \pi(x \mid \mathcal{T}^A(i))$ and $b_B(i) = \pi(y \mid \mathcal{T}^B(i) \circ b_A(i))$. Similarly, if $r_B(i+1) = r_A(i+1)+1$, Lemma 5 establishes that $\mathcal{T}^A(i)$ hasn't changed in round i, while $\mathcal{T}^B(i+1) = \mathcal{T}^B(i) \circ b_A(i) \circ b_B(i)$, which by the induction hypothesis gives Item 2. Since $b_A(i)$ and $b_B(i)$ are the correct continuations of $\mathcal{T}^A(i)$ from π, the above discussion proves that $\mathcal{T}^A(i+1)$ and $\mathcal{T}^B(i+1)$ are prefixes of $\mathcal{T}^\pi$.

Case 2: $r_B(i) = r_A(i)+1$**.** In this case, whether Bob receives an erasure or an uncorrupted message, he sets $err = 1$. Indeed, if Alice's message is not erased, then the parity Bob receives equals his saved parity (since Alice holds $r_A(i) = r_B(i)-1$ and she increases it by one in Line A.3 before sending it to Bob). In both cases Bob does not change r_B, i.e., $r_B(i+1) = r_B(i)$ and he sends the message $m_B(i) = (b_B, r_B \mod 2)$ (see Lines B.5 and B.15) from his memory. Consequently, Bob's transcript doesn't change in round i. So, $\mathcal{T}^B(i+1) = \mathcal{T}^B(i) = \mathcal{T}^A(i) \circ b_A(i-1) \circ b_B(i-1)$.

If Alice receives an erasure she sets $r_A(i+1)$ to be same as $r_A(i)$ (see Line A.12) and sets $\mathcal{T}^A(i+1) = \mathcal{T}^A(i)$. Since, $r_A(i+1) = r_A(i)$ and $r_B(i+1) = r_B(i)$ the claim holds. However, if Alice receives an uncorrupted message, she notices that $r_{rec} = (r_A(i)+1) \mod 2$ (see Line A.8) and she does not decrement r_A. In this case, $r_A(i+1) = r_A(i)+1 = r_B(i) = r_B(i+1)$ and $\mathcal{T}^A(i+1) = \mathcal{T}^A(i) \circ b_A(i) \circ b_B(i)$. Now we prove that $b_A(i) = b_A(i-1)$ and $b_B(i) = b_B(i-1)$, thus in the former case, $\mathcal{T}^B(i+1) = \mathcal{T}^A(i+1) \circ b_A(i) \circ b_B(i)$ or $\mathcal{T}^A(i+1) = \mathcal{T}^B(i+1)$. in the latter. Since in round i, Bob sets $err = 1$ we have that $m_B(i) = m_B(i-1)$ whence $b_B(i) = b_B(i-1)$. To prove that the same holds for b_A we will need the following simple claim.

▷ Claim 8. $r_A(i) = r_A(i-1)$.

Proof. Supposing round $i-1$ satisfies Item 1, we must have that $r_A(i) = r_A(i-1)$. If this were not true, then $r_B(i) = r_A(i)+1 = (r_A(i-1)+1)+1 = r_B(i-1)+2$ which is a contradiction of Lemma 5. On the other hand, if round $i-1$ satisfies Item 2, then we know r_B cannot increase in round $i-1$, whence, $r_B(i) = r_B(i-1)$. Using this we have, $r_A(i-1)+1 = r_B(i-1) = r_B(i) = r_A(i)+1$. ◁

Following the above claim, $r_A(i-1) = r_A(i)$, and we get that $\mathcal{T}^A(i-1) = \mathcal{T}^A(i)$ (Lemma 5). Therefore, $b_A(i-1) = \pi(x \mid \mathcal{T}^A(i-1)) = \pi(x \mid \mathcal{T}^A(i)) = b_A(i)$. From the induction hypothesis, $\mathcal{T}^B(i+1) = \mathcal{T}^B(i)$ is a prefix of $\mathcal{T}^\pi$. From the above discussion, we know that either $\mathcal{T}^A(i+1) = \mathcal{T}^A(i) \circ b_A(i) \circ b_B(i) = \mathcal{T}^B(i+1)$ or $\mathcal{T}^A(i+1) = \mathcal{T}^A(i)$. In the former case, it is clear that $\mathcal{T}^A(i+1)$ is a prefix of $\mathcal{T}^\pi$ whereas in the latter case, since $\mathcal{T}^A(i)$ is a prefix of $\mathcal{T}^\pi$ (by the induction hypothesis) so is $\mathcal{T}^A(i+1)$. ◀

The following lemma implies that after Alice terminates, no matter what erasures Bob sees, his values of r_B and $\mathcal{T}^B$ do not change.

▶ **Lemma 9.** *For i such that $t_A < i \leq t_B$, round i satisfies Item 1 of Lemma 7 and $r_A(i) = \frac{N}{2}$.*

Proof. Recall that $t_A < t_B$. Since Alice exits in round t_A, it must hold that at the end of this round $r_A = \frac{N}{2}$, yet, $r_A(t_A) = \frac{N}{2} - 1$ for otherwise, Alice would have terminated in the end of round $t_A - 1$. Via Lemma 7 we know that $r_B(t_A) \in \{\frac{N}{2} - 1, \frac{N}{2}\}$. If $r_B(t_A) = \frac{N}{2} - 1$ we know from the proof of case 1 in Lemma 7 that since r_A increases in round t_A, r_B also increases. In the other case, namely, when $r_B(t_A) = \frac{N}{2}$, we know from the proof of Lemma 7 (case 2) that r_B does not change, i.e., $r_B(t_A + 1) = r_B(t_A) = \frac{N}{2}$.

After Alice exits, Bob can either hear silence or an erasure and therefore for all rounds $i > t_A$, Bob sets $err = 1$ and consequently $r_B(i)$ is never incremented. It follows that for all $i > t_A$, $r_B(i) = \frac{N}{2}$. The second part of the claim follows from Corollary 6 and Lemma 7, since $\mathcal{T}^A(t_A + 1) = \mathcal{T}^B(t_A + 1)$ and since r_A, r_B do not change anymore. ◀

We are now ready to prove the main theorem and show that we can correctly simulate the noiseless protocol π under the specifications of Theorem 1. We first state Lemma 10 which will be help us bound the communication and also show that Alice eventually terminates.

▶ **Lemma 10.** *In any round $i \in [t_A]$ where there are no erasures at all, $r_A(i+1) = r_A(i) + 1$.*

Proof. Note that both parties are still active in round i. Consider the two cases of Lemma 7. If $\mathcal{T}^A(i) = \mathcal{T}^B(i)$ and both messages of round i are not erased, we showed that r_A increases (case 1). Similarly, if $\mathcal{T}^B(i) = \mathcal{T}^A(i) \circ b_A(i-1) \circ b_B(i-1)$ and no erasures occur, Alice extends her transcript and r_A increases again (case 2). ◀

▶ **Theorem 11.** *Let π be an alternating binary protocol and $T \in \mathbb{N}$ be an arbitrary integer. There exists a coding scheme Π over a 4-ary alphabet such that for any instance of Π that suffers at most T erasures overall, Alice and Bob both output $\mathcal{T}^\pi$. The simulation Π communicates at most $\mathsf{CC}(\pi) + 2T$ symbols, and has $\mathsf{CC}(\Pi) \leq 2\mathsf{CC}(\pi) + 4T$.*

Proof. Lemma 7 guarantees that at every given round, Alice and Bob hold a correct prefix of $\mathcal{T}^\pi$. Moreover, we know that by the time Alice terminates, her transcript (and hence Bob's transcript) is of length at least N, which follows from Lemma 9 and Lemma 5, i.e., from the fact that $r_A = N/2$ at termination, and that every time r_A increases by one, the length of $\mathcal{T}^A$ increases by two. Finally, note that if the number of erasures is bounded by T, then Alice will eventually reach termination because, after T erasures, r_A increases by one in every round (Lemma 10), until it reaches $N/2$ and Alice terminates.

Finally, we need to prove that the communication behaves as stated. Assume T' erasures happen up to round t_A and $T'' = T - T'$ erasures happen in rounds $[t_A + 1, t_B]$. Since every round without erasures advances r_A by one (again from Lemma 10), and since when Alice terminates we have $r_A = N/2$ (Lemma 9), then $t_A \leq T' + N/2$. Furthermore, after Alice terminates it takes one unerased (odd) round to make Bob terminate as well. Hence, at every round after t_A and until Bob terminates, Alice's silence must be erased. It follows that $t_B - t_A \leq 1 + T''$. Thus, $t_B \leq 1 + T'' + t_A \leq 1 + T + N/2$.

Every round $i \in [t_A]$ consists of two transmissions, while every round $t_A < i < t_B$ contains only a single transmission–Bob's transmission, excluding round t_B where Bob hears silence and terminates without sending a message. The total number of transmissions is then,

$$t_A + t_B - 1 \leq N + 2T' + T'' \leq N + 2T.$$

Recall that $|\Sigma| = 4$, hence $\mathsf{CC}(\Pi) \leq (N + 2T)\log 4 = 2(\mathsf{CC}(\pi) + 2T)$. ◀

3.3 Noise Resilience and Code Rate

We can compare the above result to the case where the noise is bounded as a fraction of the symbols communicated in the protocol [30, 14, 12]. In our setting, the noise amount can be arbitrary. In order to compare it to the bounded-noise model, we ask the following question. Assume an instance of Π with large amount of noise T ($T \gg N$). Then, what fraction does the noise make out of the entire communication.

As a corollary of Theorem 11 it is easy to see that the fraction of noise is lower bounded by $\frac{T}{N+2T}$ whose limit, as T tends to infinity, is $1/2$. Indeed, $1/2$ is an upper bound on the noise fraction in the bounded-noise setting [14].[4] Furthermore, if the noise is bounded to be a δ-fraction of the total communication, for some $\delta < 1/2$ then, $T \leq \delta \cdot 2t_B \leq \delta(N+2T)$ and so $T \leq \frac{\delta N}{1-2\delta}$. This implies a maximal asymptotic code rate of $1/2$. Indeed,

$$R = \frac{\mathsf{CC}(\pi)}{\mathsf{CC}(\Pi)} \geq \frac{1}{\log|\Sigma|} \cdot \frac{N}{N+2T} \geq \frac{1}{2} \cdot \frac{N}{N+2\frac{\delta N}{1-2\delta}} \geq \frac{1}{2}(1-2\delta) = \frac{1}{2} - \delta.$$

As T is unbounded relative to $\mathsf{CC}(\pi)$ and we can potentially get a zero rate, a more interesting measure is the "waste" factor, i.e., how much the communication of Π increases per single noise, for large T. In our scheme it is easy to see that each corruption delays the simulation by one round, that is, it wastes two symbols (4 bits). This implies a waste factor of 4 bits per corruption, $\omega = \lim_{T\to\infty} \frac{\mathsf{CC}(\Pi)}{T} = \frac{2N+4T}{T} = 4$.

Finally, we mention that our result extends to binary alphabet by naively encoding each symbols as two bits (this also proves the second part of Theorem 1). However, this results in a reduced tolerable noise rate of $\frac{1}{4}$. Similar to the scheme in [12], the noise resilience can be improved to $\frac{1}{3}$ by encoding each symbol via an error correcting code of cardinality 4 and distance $\frac{2}{3}$, e.g., $\{000,011,110,101\}$. In this case, two bits must be erased in order to invalidate a round. Each round (two timesteps) consists the transmission of six bits. Hence, the obtained resilience is $2/6 = 1/3$, similar to the best known resilience in the bounded-noise setting with binary alphabet [12].

References

1 Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. Distributed Computing with Channel Noise. Cryptology ePrint Archive, Report 2017/710, 2017. URL: https://eprint.iacr.org/2017/710.

2 Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. Sending a Message with Unknown Noise. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, ICDCN '18, pages 8:1–8:10, 2018. doi:10.1145/3154273.3154318.

3 Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive Protocols for Interactive Communication. In *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016. doi:10.1109/ISIT.2016.7541368.

4 Zvika Brakerski, Yael T. Kalai, and Moni Naor. Fast Interactive Coding Against Adversarial Noise. *J. ACM*, 61(6):35:1–35:30, December 2014. doi:10.1145/2661628.

5 G. Brassard, A. Nayak, A. Tapp, D. Touchette, and F. Unger. Noisy Interactive Quantum Communication. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 296–305, October 2014. doi:10.1109/FOCS.2014.39.

[4] Note that a noise level of 1/2 is also achievable for interactive coding over erasure channels in the bounded-noise setting, for alphabets of size at least 4 [12, 14]. The maximal noise for erasure channels with *binary* or *ternary* alphabet is still open.

6 M. Braverman and K. Efremenko. List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise. *SIAM Journal on Computing*, 46(1):388–428, 2017. doi:10.1137/141002001.

7 M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky. Coding for Interactive Communication Correcting Insertions and Deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, October 2017. doi:10.1109/TIT.2017.2734881.

8 Mark Braverman. Towards deterministic tree code constructions. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 161–167. ACM, 2012. doi:10.1145/2090236.2090250.

9 Mark Braverman and Anup Rao. Toward Coding for Maximum Errors in Interactive Communication. *Information Theory, IEEE Transactions on*, 60(11):7248–7255, 2014. doi:10.1109/TIT.2014.2353994.

10 Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. In *Advances in Cryptology – EUROCRYPT 2008*, pages 471–488, 2008. doi:10.1007/978-3-540-78967-3_27.

11 Varsha Dani, Thomas P. Hayes, Mahnush Movahedi, Jared Saia, and Maxwell Young. Interactive communication with unknown noise rate. *Information and computation*, 261:464–486, 2018. doi:10.1016/j.ic.2018.02.018.

12 Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal Noise in Interactive Communication Over Erasure Channels and Channels With Feedback. *IEEE Trans. Information Theory*, 62(8):4575–4588, 2016. doi:10.1109/TIT.2016.2582176.

13 Klim Efremenko, Elad Haramaty, and Yael Kalai. Interactive Coding with Constant Round and Communication Blowup. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:54, 2018. URL: https://eccc.weizmann.ac.il/report/2018/054.

14 Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal Coding for Streaming Authentication and Interactive Communication. *Information Theory, IEEE Transactions on*, 61(1):133–145, January 2015. doi:10.1109/TIT.2014.2367094.

15 R. Gelles and B. Haeupler. Capacity of Interactive Communication over Erasure Channels and Channels with Feedback. *SIAM Journal on Computing*, 46(4):1449–1472, 2017. doi:10.1137/15M1052202.

16 R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson. Explicit Capacity Approaching Coding for Interactive Communication. *IEEE Transactions on Information Theory*, 64(10):6546–6560, October 2018. doi:10.1109/TIT.2018.2829764.

17 Ran Gelles. Coding for Interactive Communication: A Survey. *Foundations and Trends® in Theoretical Computer Science*, 13(1–2):1–157, 2017. doi:10.1561/0400000079.

18 Ran Gelles and Siddharth Iyer. Interactive coding resilient to an unknown number of erasures. *CoRR*, abs/1811.02527, 2018. A preliminary full version of this paper. arXiv:1811.02527.

19 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient Coding for Interactive Communication. *Information Theory, IEEE Transactions on*, 60(3):1899–1913, March 2014. doi:10.1109/TIT.2013.2294186.

20 Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 394–403, 2014.

21 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal Error Rates for Interactive Coding I: Adaptivity and Other Settings. In *STOC '14: Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 794–803, 2014. doi:10.1145/2591796.2591872.

22 Bernhard Haeupler. Interactive Channel Capacity Revisited. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 226–235, 2014.

23 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization Strings: Channel Simulations and Interactive Coding for Insertions and Deletions. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz*

International Proceedings in Informatics (LIPIcs), pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.

24 Joseph Y. Halpern and Yoram Moses. Knowledge and Common Knowledge in a Distributed Environment. *J. ACM*, 37(3):549–587, July 1990. doi:10.1145/79147.79161.

25 IEEE. IEEE standard for Ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, pages 1–4017, March 2016. doi:10.1109/IEEESTD.2016.7428776.

26 Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC '13: Proceedings of the 45th annual ACM Symposium on theory of computing*, pages 715–724, 2013. doi:10.1145/2488608.2488699.

27 Debbie Leung, Ashwin Nayak, Ala Shayeghi, Dave Touchette, Penghui Yao, and Nengkun Yu. Capacity Approaching Coding for Low Noise Interactive Quantum Communication. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 339–352, 2018. doi:10.1145/3188745.3188908.

28 Denis Pankratov. On the Power of Feedback in Interactive Channels. [Online:] http://people.cs.uchicago.edu/~pankratov/papers/feedback.pdf, 2013.

29 Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, pages 724–733, 1992. doi:10.1109/SFCS.1992.267778.

30 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996. doi:10.1109/18.556671.

31 Leonard J. Schulman. A postcript to “Coding for Interactive Communication”. [Online:] http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt, 2003.

32 Alexander A. Sherstov and Pei Wu. Optimal Interactive Coding for Insertions, Deletions, and Substitutions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 240–251, October 2017. doi:10.1109/FOCS.2017.30.