Binary Interactive Error Resilience Beyond $1/8$ (or why $(1/2)^3 > 1/8$)

Klim Efremenko
School of Computer Science
Ben-Gurion University
Be'er Sheva, Israel
klimefrem@gmail.com

Gillat Kol, Raghuvansh R. Saxena
Department of Computer Science
Princeton University
Princeton, NJ
gillat.kol@gmail.com, rrsaxena@princeton.edu

Abstract*—Interactive error correcting codesInteractive error correcting codes** ***are codes that encode a two party communication protocol to an error-resilient protocol that succeeds even if a constant fraction of the communicated symbols are adversarially corrupted, at the cost of increasing the communication by a constant factor. What is the largest fraction of corruptions that such codes can protect against?

If the error-resilient protocol is allowed to communicate large (constant sized) symbols, Braverman and Rao (STOC, 2011) show that the maximum rate of corruptions that can be tolerated is $1/4$***. They also give a* **binary** *interactive error correcting protocol that only communicates bits and is resilient to* $1/8$ *fraction of errors, but leave the optimality of this scheme as an open problem.***

We answer this question in the negative, breaking the $1/8$ ***barrier. Specifically, we give a binary interactive error correcting scheme that is resilient to*** $5/39 > 1/8$ ***fraction of adversarial errors. Our scheme builds upon a novel construction of binary* list-decodable *interactive codes with small list size.***

***Keywords*-Interactive Coding; Error Resilience; Communication Complexity;**

I. Introduction

We study the error resilience of *binary* interactive coding schemes [1]. A *binary* interactive coding scheme with error resilience θ solves the following problem: Let Π be a two party protocol over the noiseless binary channel. Construct a protocol Π' that can simulate Π even when a θ fraction of the bits sent during Π' are adversarially corrupted. What is the largest[1] $\theta > 0$ for which there exist binary interactive coding schemes with error resilience θ?

For a related problem where the protocols are allowed to communicate symbols from a large constant sized alphabet (rather than bits), Braverman and Rao [2], building on the groundbreaking work of [1], construct a beautiful interactive coding scheme with error resilience $\frac{1}{4}$ and prove its optimality. They also show how to derive from it a binary interactive coding scheme that is resilient to $\frac{1}{8}$ fraction of errors. Since their work, it is open if there exists a binary interactive coding scheme with error resilience larger than $\frac{1}{8}$. In this work, we construct the first such scheme, answering this question from nearly a decade ago.

We note that constructing optimal binary codes is often a much more challenging task than constructing codes over large alphabets. This holds even in the non-interactive setting of error correcting codes where optimal rate *vs.* resilience trade-offs are known for codes with large alphabets (*e.g.*, the singleton bound). However, proving such trade-offs for the binary case has been a major open problem in coding theory for over half a century.

The $\frac{1}{8}$ error resilience barrier.: Not only are we lacking simulations for general binary protocols with error resilience larger than $\frac{1}{8}$, but we also do not know any such simulation for the simplest interactive task of *message exchange*, where parties wish to exchange their inputs. This is despite the fact that, for message exchange, a binary protocol with error resilience $\frac{1}{8}$ is almost obvious. Namely, both the parties simply encode their inputs with a binary error correcting code of distance $\frac{1}{2}$, and exchange these encodings[2].

In Section II, we explain why $\frac{1}{8}$ is a natural barrier for the error resilience of binary protocols, even when restricted to the message exchange task. We argue that $\frac{1}{8}$ comes from three separate $\frac{1}{2}$ factors, each stemming from a different property that we require. The first $\frac{1}{2}$ factor comes from the fact that we want a binary protocol (classical binary error correcting codes have distance approaching $\frac{1}{2}$, whereas codes over a large alphabet can have distance approaching 1 – the same factor $\frac{1}{2}$ separation is also present in the interactive regime). The second $\frac{1}{2}$ factor comes from the requirement of unique decoding (classical unique decoding can only be performed when the fraction of corruptions

Supported by the Israel Science Foundation (ISF) through grant No. 1456/18.

Supported by an Alfred P. Sloan Fellowship, the National Science Foundation CAREER award CCF-1750443, and by the E. Lawrence Keyes Jr. / Emerson Electric Co. Award.

Supported by the National Science Foundation CAREER award CCF-1750443.

[1] Actually, supremum.

[2] A distance of $\frac{1}{2}$ is the best one can get from binary error correcting codes with positive rate. This contrasts with the large alphabet case where we know error correcting codes with distance approaching 1. Correspondingly, for this case, the natural protocol for message exchange has error resilience $\frac{1}{4}$.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00051

is less than $\frac{1}{2}$, whereas the analogous threshold for list decoding is 1 – the same factor $\frac{1}{2}$ separation is also present in the interactive regime). The last $\frac{1}{2}$ factor comes from the fact that the message exchange task, like all interactive tasks, is 'two-sided', and in order to fail the task, the adversary only needs to corrupt the 'weaker' one of the two parties (*i.e.*, the one who transmits less), which needs at most $\frac{1}{2}$ the corruptions.

We show, perhaps surprisingly, that while a protocol for message exchange with any two out of the three requirements above (*i.e.*, a non-binary unique decodable interactive scheme *or* a binary list-decodable interactive scheme *or* a binary unique decodable classical scheme) can have error resilience at most $(\frac{1}{2})^2 = \frac{1}{4}$, there is a protocol satisfying all three requirements with error resilience strictly larger than $(\frac{1}{2})^3 = \frac{1}{8}$. After a lot more effort, we are also able to extend our scheme from message exchange to all interactive tasks.

A. Our Result

In this work, we show a binary interactive coding scheme with error resilience strictly greater than $\frac{1}{8}$. The following is an informal statement of our main result, a formal statement is given in Theorem VI.1.

Theorem I.1 (Informal)**.** *Let Π be a two-party binary communication protocol*[3]*. For every $\theta < \frac{5}{39}$, there exists a binary protocol Π' that simulates Π and is resilient to θ-fraction of adversarial errors. Moreover, the length of Π' is linear in the length of Π.*

Our result solves a long-standing open problem, stated explicitly in [2] (Open Problem 3), in [3] (see Version 1.3, Remark 2.1), in [4] (Open Problem 2), and in [5]. We also note that the best known upper bound on the error resilience of binary interactive coding schemes is $\frac{1}{6}$ [5]. Pinning down the optimal constant inside the range $[\frac{5}{39}, \frac{1}{6}]$ is an extremely intriguing question.

Prior to our work, it was not even known if the $\frac{1}{8}$ barrier can be crossed with a protocol Π' of *arbitrary length*. When waiving the constraint on the encoding length, the problem of finding the maximal error resilience of interactive coding schemes reduces to finding the maximal error resilience of the message exchange problem, as any communication task can be accomplished if the parties exchange their entire (possibly huge) inputs.

Binary interactive list-decodable codes.: A key ingredient in the construction of our coding scheme is a novel construction of a binary interactive *list-decodable* code with small list size. Our list-decodable scheme is resilient to $\frac{5}{32}$ fraction of errors and outputs a list of size (only) 3. We stress that our scheme in Theorem I.1 crucially relies on the list being that small, and that even a list of size 5 would not have sufficed for our result (see Section II). An imprecise statement of this result is given in Theorem I.2, a formal statement can be found in Theorem V.1.

Theorem I.2 (Informal)**.** *Let Π be a two-party deterministic binary communication protocol and let $\theta < \frac{5}{32}$. Then, there exists a binary protocol Π' such that, upon its termination, on every pair of inputs x, y for the players in Π, the first party obtains a list $L^A_{x,y}$ and the second party obtains a list $L^B_{x,y}$ with $|L^A_{x,y}|, |L^B_{x,y}| \leq 3$. The lists have the property that whenever at most θ fraction of the communication is adversarially corrupted, $\Pi(x, y) \in L^A_{x,y} \cap L^B_{x,y}$, where $\Pi(x, y)$ is the (noiseless) transcript of the execution of Π on inputs x, y. Moreover, the length of Π' is linear in the length of Π.*

We note that (in an informal sense) the parameters obtained by our interactive list-decoding scheme correspond to the best possible parameters of a (non-interactive) list-decoding scheme: There exist (non-interactive) binary codes that are list decodable with lists of size 3 from up to a $\frac{5}{16}$-fraction of errors but no more [6] (also see [7]). An extra multiplicative factor of $\frac{1}{2}$ in the error resilience is incurred as our protocol is two-way and therefore both parties need to list-decode.

B. Related Work

Since the study of coding for interactive communication was initiated by Schulman [8], [9], [1], numerous works have been published in this area [10], [2], [11], [12], [4], [13], [14], [15], [5], [16, *e.g.*]. Out of these works, the one most related to our paper is the [2] paper discussed above. For a great survey of this field, see [3].

The maximum error resilience of interactive protocols.: The question of the maximum error resilience of interactive codes, which parallels the question of maximal distance in the study of standard codes, has been one of the central topics studied by the interactive coding literature and was considered for various interactive models.

Our work, like [2] and most other works in interactive coding, assumes the "standard" (non-adaptive) model of interactive communication, where parties take turns communicating and cannot both send bits in the same round. There are models in the literature that give the parties more power and therefore have a higher resilience.

For example, the maximal error resilience question was also considered over the *adaptive* channel, where parties may collide (communicate in the same round) [17], [18], [5], [19], [20]. Over large alphabets, the error resilience of the adaptive channel was shown to be strictly higher than $\frac{1}{4}$, which is the error resilience of the non-adaptive channel [17], [19], [20]. It may be possible to obtain binary *adaptive* protocols with error resilience higher than $\frac{1}{8}$ by using the ideas in these works and losing a factor of $\frac{1}{2}$ from the result

[3]We assume, without loss of generality, that Π is deterministic, as every randomized protocol is a distribution over deterministic protocols.

for large alphabets. However, our work is the first one to show that one does not necessarily have to lose this factor of $\frac{1}{2}$. Consequently, we suspect that our techniques in this paper, in combination with some of our tools in [19], [20], can save us from losing a factor of $\frac{1}{2}$ in those models as well.

Error resilience better than $\frac{1}{8}$ for the binary case was proved for the channel with noiseless *feedback* as well. The work of [5] gives an interactive coding scheme over the non-adaptive binary channel with feedback with error resilience $\frac{1}{6}$, and an interactive coding scheme over the adaptive binary channel with feedback with error resilience $\frac{1}{3}$. Both of these results are shown to be tight [5]. The maximum interactive error resilience of the erasure channel and the insertions and deletions channel was studied in [5], [21], [22], [23].

Interactive list-decoding.: The notion of list-decodable codes was extended to the interactive setting by [4], [17], [24]. In [4], it is shown that for every $\varepsilon > 0$, there exists an interactive list-decoding scheme over a large constant-sized alphabet with a list of size $\mathrm{poly}(\frac{1}{\varepsilon})$ and error tolerance $\frac{1}{2} - \varepsilon$, that only blows-up the communication linearly[4]. Computationally efficient interactive list-decoding schemes with similar guarantees are constructed in [17], [24][5].

Open Problem 2 in [4] remarks that "one can modify all our protocols to work over a binary channel with a loss of a factor of two in the error rates that one can handle". This modification seems to require non-trivial effort and would imply binary interactive list-decodable codes with a list of size $\mathrm{poly}(\frac{1}{\varepsilon})$ and error tolerance $\frac{1}{4} - \varepsilon$. While the error tolerance guarantee given by this modification is better than the error tolerance promised in Theorem I.2 ($\frac{1}{4}$ *vs.* $\frac{5}{32}$), it does not suffice for our purposes as the list size (while still constant) is too big.

Shayevitz [25] studied (standard) list-decodable codes in the presence of noiseless feedback, showing that feedback can in fact improve the parameters of list-decodable codes. The noiseless feedback model used in [25] is incomparable to our interactive setting.

C. Techniques

Recall that the [2] protocol for the binary alphabet has error resilience $\frac{1}{8}$, and this value stems from the fact that the *optimal* decoding radius (error resilience) of binary error correcting codes is $\frac{1}{4}$. At a high level, our protocol gets higher error resilience by artificially implementing binary codes with error resilience better than the optimal $\frac{1}{4}$. We next give a very simplified description of our protocol.

[4]We note that the work of [4] actually gives much stronger statements: It shows that interactive list-decoding with a list of size $\mathrm{poly}(\frac{1}{\varepsilon})$ is possible as long as $\alpha+\beta < 1-\varepsilon$, where α and β are the fractions of the communication from Alice to Bob and from Bob to Alice (respectively) corrupted by the adversary.

[5]The list decoding scheme in [24] is of constant rate, improving over the polynomially-small rate obtained by [17].

At the beginning of the protocol, the parties use a binary interactive list-decodable code with super small (size 3) lists that tolerates $\frac{5}{16} > \frac{1}{4}$ fraction of errors, which we construct for this purpose (see Subsection I-A). After this phase, both parties have small lists of candidates for the correct output, and Bob sends his list to Alice. Since Alice knows both her list and Bob's list, she can output correctly. She then sends the index of the correct output in Bob's list to Bob. The reason Alice sends the index instead of sending the actual output itself is that the index can only take one of 3 values. While no (asymptotic) binary error correcting code has distance greater than $\frac{1}{2}$, binary codes with up to 4 codewords can have distance $\frac{2}{3}$. We use such a code for the index and exploit its larger distance to get higher error resilience overall.

To conclude, the saving in our protocol stems from a novel combination of two binary error correcting codes with "better-than-optimal" error resilience, namely, a binary list-decodable code with super short list size and a binary error correcting code with few codewords. A more detailed overview, highlighting some of the challenges that the implementation of this high-level idea entails, is found in Section II. We believe that this high-level idea can guide the design of additional interactive codes.

II. Overview of Our Protocol

In this section, we gradually build various aspects of our simulation scheme, highlighting the roles they play.

A. [2] And The Message Exchange Problem

The work most directly related to our work is by Braverman and Rao [2]. In this work, the authors show that, for any $\theta < \frac{1}{4}$, any noiseless two party communication protocol can be simulated over a channel that can adversarially corrupt any θ fraction of the symbols communicated over the channel. Moreover, the authors also show that the parameter $\frac{1}{4}$ is the largest possible for which such a claim holds, as long as the channel is non-adaptive, thereby showing that the *maximal error resilience* of the non-adaptive channel is $\frac{1}{4}$.

[2] provides a simulation for arbitrary noiseless protocols over the noisy two party channel. Nonetheless, it is worthwhile to consider this simulation for the basic *message exchange* protocol, where both parties have a private input that they want to share with each other. Considering only this simple task already provides a simulation scheme for *all* tasks with an exponential blowup, as any communication task can be performed by the parties simply exchanging their (potentially exponentially large) inputs. Thereafter, comes the harder task of reducing the length of the simulation and making it linear in the length of the original protocol.

We incorporate this high level blueprint into our sketch and first illustrate the subset of ideas needed even if one only wants to simulate the message exchange task in

Subsection II-B and Subsection II-C, and then build on these ideas to get a simulation scheme for all noiseless protocols (with a constant blowup) in Subsection II-D.

B. The $\frac{1}{8}$ Barrier For Message Exchange

In the binary regime, the best known error resilience of an interactive coding scheme is $\frac{1}{8}$ [2]. This even holds for the restricted task of message exchange. The value $\frac{1}{8}$ seems to be a natural barrier due to the following three reasons, each of which contributes a multiplicative factor of $\frac{1}{2}$.

- **Binary alphabet:** It is well known that the maximum distance of binary error correcting codes is $\frac{1}{2}$, whereas the maximum distance of *general* error correcting codes (*i.e.*, with a constant sized alphabet) can be arbitrarily close to 1. This factor of $\frac{1}{2}$ separation is also found in the best known distance parameters for binary and general *tree codes* (see for example the construction in [1]).
  As error correcting codes and tree codes are commonly used as 'building blocks' in interactive coding schemes, a factor of $\frac{1}{2}$ separation between binary and general coding schemes is found in many works. Indeed, [2] is one such work.
- **Unique decoding:** Another factor of $\frac{1}{2}$ in the error-resilience comes from the fact that if the minimum distance between two codewords of an error correcting code is δ, then the parties can decode to a unique codeword only if the number of errors is at most $\frac{\delta}{2}$. Going beyond this threshold requires working in the list-decoding regime where the parties output a small list of values that is guaranteed to contain the correct codeword. In the list decoding regime, the fraction of errors that one can decode from is double that in the unique decoding regime, and can be made arbitrarily close to 1 (using a constant sized alphabet).
- **Two-way task:** The last factor of $\frac{1}{2}$ is due to the fact that the message exchange task is successful only if *both* the parties output each others' input. In other words, in order to derail the message exchange task, the adversary only needs to make sure that one of the parties outputs incorrectly. In particular, the adversary can invest all his errors on the party that speaks less. Since this party talks in at most half the rounds, the fraction of corruptions required to derail the protocol is lower by a factor of $\frac{1}{2}$.

The fact that we have three requirements, each of which hurt the error resilience by a factor of $\frac{1}{2}$, is not strong reason to believe that the three requirements together imply an error resilience of at most $\frac{1}{8}$. Indeed, it is possible that there are ways to combine these requirements so that the resulting error resilience is higher. The reason $\frac{1}{8}$ is a natural barrier for the error resilience is that combining any two of the above requirements (and not the third) implies an error resilience of at most $\frac{1}{4}$. Indeed, we have the following bounds:

- **No binary alphabet:** Suppose that one relaxes the binary alphabet constraint but still requires unique decoding and two-way message exchange. As mentioned above, [2] show that the error resilience of their simulation is optimal in this case and no protocol can get error resilience higher than $\frac{1}{4}$.
- **No unique decoding:** Suppose now that one relaxes the unique decoding constraint (and allows each party to output a small set of guesses for the input of the other party, rather than a single value), but still requires binary alphabet and two-way message exchange. For this setting, [4] give a protocol that has error resilience $\frac{1}{4}$[6]. It seems to be folklore that one cannot have a protocol with higher error resilience, but as we could not find a proof, we provide an informal proof of this fact in Section A.
- **No two-way task:** Finally, consider the setting where only one party is required to send its input to the other party using a binary alphabet and the other party needs to uniquely decode this input. This is the familiar setting of binary error-correcting codes. It is well-known that in this setting, the maximum error resilience is $\frac{1}{4}$ and is achieved, for instance, by random codes.

Having shown the significance of the $\frac{1}{8}$ barrier, we next describe our protocol and illustrate how it surpasses this barrier.

C. Our Message Exchange Protocol

Our protocol for the message exchange problem will have $39N$ rounds and error resilience $\frac{5}{39} > \frac{1}{8}$ (for an N linear in the size of the parties' inputs). This means that the adversary can corrupt less than $\frac{5}{39} \cdot 39N = 5N$ rounds.

Our protocol consists of three phases of different lengths. The first phase and the third phase will have lengths $16N$ and $3N$ respectively and in these phases, Alice will be sending messages to Bob. The second phase will have length $20N$ where Bob will be sending messages to Alice. The protocol is given in Algorithm 1, illustrated in Figure 1, and described below.

The protocol follows the following high level scheme (see Figure 1): In phase 1, Alice will send her input x^A to Bob encoded using a *list-decodable* error correcting code. We choose an error correcting code that can decode from up to a $\frac{5}{16}$ fraction of corruptions using a list of size 3. As the length of this phase is $16N$ and the number of corruptions is less than $5N$, using such a code allows Bob to compute a set L of values, $|L| \leq 3$, that is guaranteed to have Alice's input.

In phase 2, Bob will send his input x^B to Alice along with the set L that he computed in phase 1, encoded using

[6]More precisely, they give a protocol with error resilience $\frac{1}{2}$ with a constant sized alphabet but remark that their ideas can be extended to the binary regime at the cost of another $\frac{1}{2}$ in the error resilience.

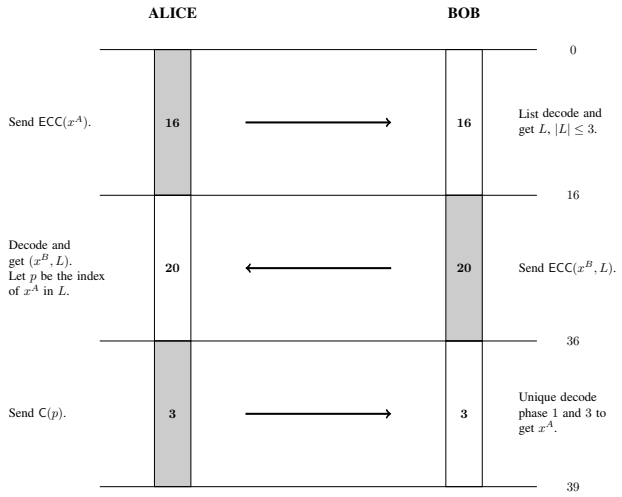

Figure 1. The 3 stages in our $\frac{5}{39}$ error resilient protocol (lengths not to scale). Here, ECC is a binary error correcting code with distance $\frac{1}{2}$ and list decodable from up to $\frac{5}{16}$-fraction of errors with a list of size 3 and C is a binary error correcting code with only 3 codewords and distance $\frac{2}{3}$. It is well known that such codes exist. In fact, codes with 4 codewords and distance $\frac{2}{3}$ also exist (see Lemma IV.2).

a *uniquely decodable* error correcting code[7]. As the length of this phase is $20N$ and the number of corruptions is less than $5N = \frac{1}{4} \cdot 20N$, there exist codes that allow Alice to decode x^B and L uniquely from Bob's message. Thus, after phase 2, Alice knows both her output and the set L that Bob computed in phase 1.

In the third phase, Alice will help Bob identify x^A inside the set L. As Alice knows both x^A and L, after phase 2, she can do this by simply sending the index of x^A inside L in this phase (of course, encoded using an error correcting code). However, binary error correcting codes have distance of at most $\frac{1}{2}$, implying that the distance obtained by Alice in phase 1 and phase 3 add up to $\frac{1}{2} \cdot (16N + 3N) = 9.5N$. As $9.5N < 2 \cdot 5N$, an adversary that can corrupt less than $5N$ messages can make Bob receive the same transcript for two different inputs for Alice.

Thus, it seems that in order for Bob to decode Alice's input, Alice somehow needs to encode her message in phase 3 using a binary code of distance $> \frac{1}{2}$[8]. However, as stated in Subsection II-B, binary error correcting codes have distance at most $\frac{1}{2}$. How is it possible to attain the unattainable?

Attaining the unattainable.: It turns out that there is one setting where binary error correcting codes can get distance strictly larger than $\frac{1}{2}$, and this is the setting of error correcting codes with few codewords (as opposed to asymptotic codes that are being used regularly). As an extreme example, consider the case where one wishes to have an error correcting code with only two codewords. In this case, it is actually possible to get a code with distance 1! Indeed, simply consider the code with the codewords $00\cdots0$ and $11\cdots1$.

In fact, binary codes with distance exceeding $\frac{1}{2}$ are known to exist for any constant number of codewords, with the distance approaching $\frac{1}{2}$ as the number of codewords increases. As all we need in phase 3 is a binary code with 3 codewords (recall that $|L| \leq 3$ and Alice only wants to encode an index $\leq |L|$), codes with few codewords are the way to go! We next employ the fact that there exists a code with 4 codewords that has distance $\frac{2}{3}$ (see Lemma IV.2 for an explicit construction) to get that the combined distance obtained by Alice in phase 1 and phase 3 is $\frac{1}{2} \cdot 16N + \frac{2}{3} \cdot 3N = 10N$. As the budget of the adversary is less than $5N$, this ensures that Bob can always decode Alice's input from the bits received by him.

Below, we present a slightly more formal description of our protocol and analysis. Before the description however, we briefly state why our protocol does not run into the barriers described in Subsection II-B.

The first barrier described in Subsection II-B was that binary error correcting codes can only offer a distance of $\frac{1}{2}$. We get around this barrier by using codes with few codewords that offer better distance guarantees. We also note that, for larger alphabets, codes with few codewords offer the same distance guarantees as asymptotic codes (both have distance close to 1) and therefore, a similar trick cannot be used to improve the error resilience of, say, [2].

The second barrier in Subsection II-B was that message exchange requires unique decoding and unique decoding can only decode from half of the distance guaranteed by the code. We get around this barrier by using list decoding to a super small (size 3) list in phase 1. Although less powerful than full-fledged list-decoding (where lists of any constant size are allowed), even size 3 list-decoding is powerful enough to get around the $\frac{1}{2}$ barrier that unique decoding runs into.

Finally, the last barrier in Subsection II-B was that two-way decoding should (roughly) have half the error resilience of one-way decoding as maybe it can be split into two one-way schemes (one for Alice and one for Bob). We do not run into this barrier as in our scheme, Alice needs to know Bob's list in order to send him the index. Since this is not possible without Bob communicating, our scheme cannot be split into two one-way schemes, one for each party.

As is evident from the discussion, all three barriers interact to allow the saving over $1/8$.

Analyzing our message exchange protocol.: We describe our message exchange protocol more formally in Algorithm 1. Our protocol has a total of $39N$ rounds,

[7]Note that this is the only phase where Bob speaks in our protocol, and therefore Alice must unique decode in this phase.

[8]We can also make phase 3 longer, but it is easily seen that the same arguments apply as long as phase 3 is less than $4N$ rounds. If phase 3 has length at least $4N$, then the error resilience is at most $\frac{5N}{16N+20N+4N} = \frac{1}{8}$ and does not give us our result.

where N is chosen to be large enough so that all the codes mentioned in Algorithm 1 exist. It is well known that this can be achieved by an N that is linear in the length of Alice's and Bob's inputs.

Algorithm 1 Our $\frac{5}{39}$ error resilient protocol simplified for the message exchange task.

Phase 1:

1: Alice sends $\mathsf{ECC}(x^A)$ to Bob in the first $16N$ rounds. Here, ECC is a binary error correcting code with distance $\frac{1}{2}$ that is list-decodable from up to $\frac{5}{16}$ fraction of errors with a list of size 3.
2: Bob receives $\rho \in \{0,1\}^{16N}$ from Alice and decodes ρ to get a list L of size at most 3.

Phase 2:

3: Bob sends $\mathsf{ECC}(x^B, L)$ to Alice in the next $20N$ rounds. Recall that ECC is a binary error correcting code with distance $\frac{1}{2}$ and therefore uniquely decodable from up to $\frac{1}{4}$ fraction of errors.
4: Alice unique decodes Bob's message to get x^B and L. She outputs x^B and sets $\mathsf{p} \in [3]$ to be the index of x^A in L.

Phase 3:

5: Alice sends $\mathsf{C}(\mathsf{p})$ to Bob in the final $3N$ rounds. Here, C is a binary error correcting code with only 3 codewords and distance $\frac{2}{3}$. It is well known that such codes exist.
6: Bob receives $\tau \in \{0,1\}^{3N}$ and outputs L_{q} where q minimizes $\delta(\mathsf{q}) = \Delta\left(\mathsf{ECC}(L_{\mathsf{q}}), \rho\right) + \Delta\left(\mathsf{C}(\mathsf{q}), \tau\right)$. Here, $\Delta(\cdot,\cdot)$ denotes the Hamming distance between strings and L_{q} is the q^{th} element in L.

We now argue why, at the end of Algorithm 1, if the number of corruptions is less than $\frac{5}{39} \cdot 39N = 5N$, then both Alice and Bob can output each others' inputs. The reason Alice can output Bob's input x^B is because, in Line 3 of phase 2, Bob sends x^B encoded using an error correcting code that is uniquely decodable from up to $\frac{1}{4}$ errors. As the length of phase 2 is $20N$ rounds and the number of errors is $< \frac{1}{4} \cdot 20N = 5N$, Alice will be able to output x^B (and recover L) correctly.

As Alice can compute L correctly, the correctness of Bob's output follows if we can show that $\mathsf{p} = \mathsf{q}$. For this we first observe that $\delta(\mathsf{p})$ is at most the number of corruptions and is therefore, strictly less than $5N$. As q is chosen to the minimizer of $\delta(\cdot)$, we also have that $\delta(\mathsf{q}) < 5N$ and $\delta(\mathsf{p}) + \delta(\mathsf{q}) < 10N$. Thus, in order to show that $\mathsf{p} = \mathsf{q}$, it is enough to show that $\delta(\mathsf{p}) + \delta(\mathsf{p}') \geq 10N$ for all $\mathsf{p} \neq \mathsf{p}'$. This follows simply using the triangle inequality and the distance property of our codes as:

$$\delta(\mathsf{p}) + \delta(\mathsf{p}') \geq \Delta\left(\mathsf{ECC}(L_{\mathsf{p}}), \mathsf{ECC}(L_{\mathsf{p}'})\right) + \Delta\left(\mathsf{C}(\mathsf{p}), \mathsf{C}(\mathsf{p}')\right)$$

$$\geq \frac{1}{2} \cdot 16N + \frac{2}{3} \cdot 3N = 10N,$$

where the last inequality is because the distance of ECC is $\frac{1}{2}$ and the distance of C is $\frac{2}{3}$.

D. Extending to Interactive Coding

Building a protocol that solves the message exchange task with error resilience $\frac{5}{39} > \frac{1}{8}$ is only the first step in our general simulation. In fact, this first step was almost trivial in the work of [2]. It remains to extend the simulation to cover all possible noiseless protocols. This extension has several challenges that are outlined below.

Binary interactive list-decodable codes with a list of size 3.: First and foremost comes the challenge that the list-decodable codes used in phase 1 of our protocol for message exchange have no analogue for general interactive tasks. In fact, the only list-decodable interactive error correcting codes for general interactive tasks are the ones described in [4], [24]. These are too weak for us due to three reasons: (1) Firstly, both of the works [4] and [24] work with a large constant sized alphabet and not the binary alphabet that we work in. (2) Secondly, both of these works ignore constants in the list size they obtain. For us, however, this constant is closely connected to the distance of the codes we can use in phase 3 and therefore directly affects the maximum error resilience we can obtain. (3) Lastly, both of these works do not give the distance properties we need from our list-decodable code in phase 1. In other words, for these works, we do not have a guarantee that any two 'codewords' have a distance of $\frac{1}{2}$ as was needed in our analysis.

A lot of technical work goes into getting list-decodable interactive error correcting codes with the properties listed above. We do this in Section V where we build upon the ideas in [4] to get a list-decodable interactive code with the desired properties. An important ingredient in this protocol is our notion of 'boosted list-decodable tree codes' that provide stronger guarantees than standard list-decodable tree codes and will be crucial in ensuring the above properties. We define and show the existence of these tree codes in Subsection IV-B and Subsection IV-C.

We also mention that parameters of the list-decodable interactive codes we construct "correspond" to the optimal parameters in the non-interactive setting in the following sense: In the non-interactive setting, it is well known [6] that it is not possible to decode from more than $\frac{5}{16}$ errors with a list of size 3. Our list decodable interactive codes decode from up to $\frac{5}{32}$ errors in using a list of size 3. This parameter corresponds to the one for classical codes, up to a factor of $\frac{1}{2}$ that we lose as in our case, both parties need to decode.

Adding a fourth codeword.: Our task is not complete even after building the aforementioned list-decodable interactive codes as they turn out to be incompatible with Algorithm 1 in the following sense. Any interactive code

that has a sub-exponential blowup requires Alice and Bob to interact. To allow this interaction, we must have Bob send messages to Alice in phase 1 of our scheme when they are running the list decodable interactive code. Consequently, phase 1 of our protocol will be longer, which means that in order to not lose too much in the error resilience, phase 2 of our protocol will have to be much shorter.

However, if phase 2 of our protocol is shorter, then Alice is not guaranteed to decode Bob's list L correctly, which means that the index p that Alice computes may not be the index of the correct output in Bob's actual list, and everything that Alice sends in phase 3 may be meaningless!

To fix this problem, we first observe that in case Alice does not decode L correctly in phase 2, then there must have been many errors in phase 2. As the total number of errors is limited, it means that there must have been relatively fewer errors in phase 1. We ensure that this number is small enough so that the 'most likely' output in phase 1 is the correct one. Additionally, we add an extra codeword to phase 2 that Alice sends when she thinks that phase 2 had many corruptions.

If Bob decodes to this extra codeword in phase 3, then he understands this as being a signal that there were many errors in phase 2 (or in phase 3) and he should simply output the 'most likely' outcome in phase 1. Otherwise, Bob decodes as usual. Finally, as there exists a binary code with 4 codewords and distance $\frac{2}{3}$, this does not affect the error resilience of our protocol.

III. Preliminaries and Formal Problem Definition

Our proof uses the following version of the Chernoff bound.

Lemma III.1 (Multiplicative Chernoff bound)**.** *Suppose $X_1, \cdots, X_n$ are independent random variables taking values in $\{0,1\}$. Let X denote their sum and let $\mu = \mathbb{E}[X]$ denote the sum's expected value. Then,*

$$\Pr[X \geq (1+\delta)\mu] \leq \mathrm{e}^{-\frac{\delta^2\mu}{3}}, \qquad \forall 0 < \delta < 1,$$
$$\Pr[X \leq (1-\delta)\mu] \leq \mathrm{e}^{-\frac{\delta^2\mu}{2}}, \qquad \forall 0 < \delta < 1.$$

Definition III.2. *Let $n > 0$ and $s, t \in \{0,1\}^n$. Define the suffix distance, $\delta_{\mathsf{suf}}(s,t)$, between s and t as*

$$\delta_{\mathsf{suf}}(s,t) = \max_{i \in [n]} \frac{\Delta(s_{\geq i}, t_{\geq i})}{n-i+1}.$$

For $r > 1$, we extend the definition of suffix distance to strings $s, t \in (\{0,1\}^r)^n$ as follows: If $s = s_1 s_2 \cdots s_n$ where $s_i \in \{0,1\}^r$ for $i \in [n]$, then, let $s' \in \{0,1\}^{rn}$ denote the string $s' = s_1 \| s_2 \| \cdots \| s_n$. Define t' similarly. We define:

$$\delta_{\mathsf{suf}}(s,t) = \delta_{\mathsf{suf}}(s', t').$$

A. The Binary Two Party Communication Model

We now formally define the binary two party communication model.

A (deterministic) protocol $\Pi = \{T, p, \mathcal{X}^C, \mathcal{Y}^C, f^C, \mathsf{out}^C\}_{C \in \{A,B\}}$ in the binary two party communication model is defined by a length parameter T, an order of turns given by a sequence $p \in \{A,B\}^T$, input sets $\mathcal{X}^A$ and $\mathcal{X}^B$, output sets $\mathcal{Y}^A$ and $\mathcal{Y}^B$, transmission functions f^A and f^B, and output functions out^A and out^B. Here, for $C \in \{A,B\}$, the functions f^C and out^C are of the types:

$$f^C : \mathcal{X}^C \times \{0,1\}^{<T} \to \{0,1\},$$
$$\mathsf{out}^C : \mathcal{X}^C \times \{0,1\}^T \to \mathcal{Y}^C.$$

Such a protocol Π is executed is the presence of an adversary. We first define an adversary Adv for Π and then define an execution of Π in the presence of Adv. An adversary Adv for Π is defined by two functions Adv^A and Adv^B of the types:

$$\mathsf{Adv}^A, \mathsf{Adv}^B : \mathcal{X}^A \times \mathcal{X}^B \to \{0,1\}^T.$$

An execution of Π in the presence of Adv proceeds as follows: At the beginning of the execution, Alice and Bob start with inputs $x^A \in \mathcal{X}^A$ and $x^B \in \mathcal{X}^B$ respectively. The execution consists of T rounds and before the i^{th} rounds, for $i \in [T]$, Alice and Bob have transcripts $\pi^A, \pi^B \in \{0,1\}^{i-1}$ respectively. In round i, if $p_i = A$, then Alice transmits the symbol $f^A(x^A, \pi^A)$ while Bob receives the symbol $\mathsf{Adv}_i^B(x^A, x^B)$. Both the parties add these symbols to π^A and π^B respectively. Similarly, if $p_i = B$, then Bob transmits the symbol $f^B(x^A, \pi^B)$ while Alice receives the symbol $\mathsf{Adv}_i^A(x^A, x^B)$. Both the parties add these symbols to π^A and π^B respectively. After T such rounds, Alice and Bob output $\mathsf{out}^A(x^A, \pi^A)$ and $\mathsf{out}^B(x^B, \pi^B)$ respectively.

Observe that this execution, and therefore π^A, π^B are completely determined by x^A, x^B, Π, and Adv. We shall often use $\mathsf{out}^A_{\Pi,\mathsf{Adv}}(x^A, x^B)$ to denote $\mathsf{out}^A(x^A, \pi^A)$ and $\mathsf{out}^B_{\Pi,\mathsf{Adv}}(x^A, x^B)$ to denote $\mathsf{out}^B(x^B, \pi^B)$.

Corruptions.: Consider an execution of Π in the presence of the adversary Adv. For $R \subseteq [T]$, we define the number of corruptions in the rounds in R to be

$$\mathsf{corr}_{\Pi,\mathsf{Adv},R}(x^A, x^B) = \sum_{i \in R} \mathbb{1}\left(\pi_i^A \neq \pi_i^B\right).$$

Recall that π^A, π^B are completely determined by x^A, x^B, Π, and Adv and therefore the corr is well defined. For $i \in [T]$, we use $\mathsf{corr}_{\Pi,\mathsf{Adv},i}(x^A, x^B)$ to denote $\mathsf{corr}_{\Pi,\mathsf{Adv},\{i\}}(x^A, x^B)$, $\mathsf{corr}_{\Pi,\mathsf{Adv},\leq i}(x^A, x^B)$ to denote $\mathsf{corr}_{\Pi,\mathsf{Adv},[i]}(x^A, x^B)$, $\mathsf{corr}_{\Pi,\mathsf{Adv},>i}(x^A, x^B)$ to denote $\mathsf{corr}_{\Pi,\mathsf{Adv},\{i+1,i+2\cdots,T\}}(x^A, x^B)$, *etc.* Finally, we omit the subscript R when $R = [T]$.

Noiseless adversary.: Observe that for any protocol Π, there is a unique adversary Adv that satisfies $\mathsf{corr}_{\Pi,\mathsf{Adv},\leq T}(x^A, x^B) = 0$ for all $x^A \in \mathcal{X}^A$ and $x^B \in \mathcal{X}^B$. We call this adversary the noiseless adversary and denote it by Adv^*. It follows that when Π is executed in the presence of Adv^* and the inputs are x^A and x^B respectively, then we have $\pi^A = \pi^B$. We use $\Pi(x^A, x^B)$ to denote this common value.

IV. Results From Coding Theory

A. Error Correcting Codes

We will need the following standard result concerning error correcting codes.

Lemma IV.1. *For all $\epsilon > 0$, there exists an integer $n_0 > 0$ such that for all $n \geq n_0$, there exists a function $\mathsf{ECC}_{n,\epsilon} : \{0,1\}^{\lfloor \frac{\epsilon^2 \cdot n}{10} \rfloor} \to \{0,1\}^n$ such that for all $s \neq t \in \{0,1\}^n$, we have*

$$\Delta\left(\mathsf{ECC}_{n,\epsilon}(s), \mathsf{ECC}_{n,\epsilon}(t)\right) \geq \left(\frac{1}{2} - \epsilon\right) \cdot n.$$

We also use the following well known lemma concerning codes with 4 codewords.

Lemma IV.2. *For every $n > 0$, there exists a function $\mathsf{C}_n : \{0,1,2,3\} \to \{0,1\}^{3n}$ such that for all $i \neq i' \in \{0,1,2,3\}$, we have*

$$\Delta\left(\mathsf{C}_n(i), \mathsf{C}_n(i')\right) = 2n.$$

Proof: Let str^z denote the string obtained by concatenating str to itself z times, *e.g.*, $(934)^5 = 934934934934934$. We define the function C_n as follows:

$$\mathsf{C}_n(i) = \begin{cases} (000)^n & , i = 0 \\ (011)^n & , i = 1 \\ (101)^n & , i = 2 \\ (110)^n & , i = 3 \end{cases}.$$

The lemma then follows straightforwardly. ∎

B. List-Decodable Tree Codes with Binary Alphabet

Throughout this section, we fix Σ be a non-empty finite set. Let $S \subseteq \Sigma^*$ be a set of strings over Σ. We define $\mathsf{pre}(S)$ to be the set of all prefixes of all strings in S, *i.e.* the set

$$\mathsf{pre}_i(S) = \{s_{\leq i} \mid s \in S, i \leq |s|\}.$$
$$\mathsf{pre}(S) = \bigcup_{i>0} \mathsf{pre}_i(S).$$

For $r > 0$ and a function $f : \Sigma^* \to \{0,1\}^r$, we use $\overline{f} : \Sigma^* \to (\{0,1\}^r)^*$ to denote the function that takes $s \in \Sigma^*$ to an $|s|$-length string over $\{0,1\}^r$ such that the i^{th} coordinate of $\overline{f}(s)$, for $i \in [|s|]$, is $f(s_{\leq i})$. Next, if $r, \alpha > 0$, $f : \Sigma^* \to \{0,1\}^r$ is a function, and $\tilde{s} \in (\{0,1\}^r)^*$ is a string, we define

$$\mathsf{near}^f_{r,\alpha,i}(\tilde{s}) = \{s \in \Sigma^i \mid \delta_{\mathsf{suf}}(\overline{f}(s), \tilde{s}_{\leq i}) < \alpha\}.$$

$$\mathsf{near}^f_{r,\alpha}(\tilde{s}) = \bigcup_{i \in [|\tilde{s}|]} \mathsf{near}^f_{r,\alpha,i}(\tilde{s}).$$

Also, define:

$$\delta^f_{\mathsf{pre}}(\mathsf{near}^f_{r,\alpha}(\tilde{s}), \tilde{s}) = \sum_{i \in [|\tilde{s}|]} \sum_{s \in \mathsf{pre}_i(\mathsf{near}^f_{r,\alpha}(\tilde{s}))} \Delta(f(s), \tilde{s}_i).$$

The following lemma captures what we need from these definitions.

Lemma IV.3. *For all $r, \alpha > 0$, $\tilde{s} \in (\{0,1\}^r)^*$ and $f : \Sigma^* \to \{0,1\}^r$, we have that*

$$\delta^f_{\mathsf{pre}}(\mathsf{near}^f_{r,\alpha}(\tilde{s}), \tilde{s}) \leq r\alpha \cdot |\mathsf{pre}(\mathsf{near}^f_{r,\alpha}(\tilde{s}))|.$$

Proof: Consider, for $i > 0$, the function $g_i : \mathsf{pre}_i(\mathsf{near}^f_{r,\alpha}(\tilde{s})) \to \mathsf{near}^f_{r,\alpha}(\tilde{s})$ that takes $s' \in \mathsf{pre}_i(\mathsf{near}^f_{r,\alpha}(\tilde{s}))$ to the lexicographically smallest $s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})$ such that s' is a prefix of s. Owing to the definition of pre, at least one such s always exists and therefore g_i is well-defined. Furthermore, observe that g_i is an injection. We get the following equalities:

$$\begin{aligned} \delta^f_{\mathsf{pre}}(\mathsf{near}^f_{r,\alpha}(\tilde{s}), \tilde{s}) &= \sum_{i \in [|\tilde{s}|]} \sum_{s \in \mathsf{pre}_i(\mathsf{near}^f_{r,\alpha}(\tilde{s}))} \Delta(f(s), \tilde{s}_i) \\ &= \sum_{i \in [|\tilde{s}|]} \sum_{s \in \mathsf{im}(g_i)} \Delta(f(s_{\leq i}), \tilde{s}_i) \end{aligned} \tag{1}$$

$$\begin{aligned} |\mathsf{pre}(\mathsf{near}^f_{r,\alpha}(\tilde{s}))| &= \sum_{i \in [|\tilde{s}|]} |\mathsf{pre}_i(\mathsf{near}^f_{r,\alpha}(\tilde{s}))| = \sum_{i \in [|\tilde{s}|]} |\mathsf{im}(g_i)| \\ &= \sum_{i \in [|\tilde{s}|]} \sum_{s \in \mathsf{im}(g_i)} 1. \end{aligned} \tag{2}$$

To proceed, we note from the definition of g_i, that if $s \in \mathsf{im}(g_i)$ for some $i > 0$, then $s \in \mathsf{im}(g_{i'})$ for all $i \leq i' \leq |s|$. Using $i_*(s)$ to denote the smallest i such that $s \in \mathsf{im}(g_i)$ and defining $i_*(s) = |s| + 1$ if no such i exists, we get:

$$\begin{aligned} &\delta^f_{\mathsf{pre}}(\mathsf{near}^f_{r,\alpha}(\tilde{s}), \tilde{s}) \\ &= \sum_{i \in [|\tilde{s}|]} \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} \mathbb{1}\left(s \in \mathsf{im}(g_i)\right) \cdot \Delta(f(s_{\leq i}), \tilde{s}_i) \\ &\qquad \text{(Equation 1)} \\ &= \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} \sum_{i \in [|\tilde{s}|]} \mathbb{1}\left(s \in \mathsf{im}(g_i)\right) \cdot \Delta(f(s_{\leq i}), \tilde{s}_i) \\ &= \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} \sum_{i=i_*(s)}^{|\tilde{s}|} \Delta(f(s_{\leq i}), \tilde{s}_i) \\ &\leq \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} r \cdot (|s| + 1 - i_*(s)) \cdot \delta_{\mathsf{suf}}\left(\overline{f}(s), \tilde{s}_{\leq |s|}\right) \\ &\leq \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} r\alpha \cdot (|s| + 1 - i_*(s)) \quad (\text{As } s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})) \end{aligned}$$

$$= r\alpha \cdot \sum_{s \in \mathsf{near}^f_{r,\alpha}(\tilde{s})} \sum_{i=i_*(s)}^{|\tilde{s}|} 1$$
$$= r\alpha \cdot |\mathsf{pre}(\mathsf{near}^f_{r,\alpha}(\tilde{s}))|. \qquad \text{(Equation 2)}$$

■

We are now ready to define list tree codes.

Definition IV.4 (List-Decodable Tree Codes with Binary Alphabet)**.** *Let $r, L, \alpha > 0$ and $f : \Sigma^* \to \{0,1\}^r$ be a function. We say that f is an (r, L, α)-list tree code if for all $\tilde{s} \in (\{0,1\}^r)^*$, we have:*

$$|\mathsf{pre}(\mathsf{near}^f_{r,\alpha}(\tilde{s}))| < L \cdot |\tilde{s}|.$$

C. Boosted List Tree Codes

It turns out that the above notion of list tree codes will not be sufficient for our needs and we need to boost it to get stronger guarantees. We next define these boosted tree codes. Define the separation function $\mathsf{sep} : \{0,1,2,3,4\} \to \mathbb{R}$ as follows:

$$\mathsf{sep}(i) = \begin{cases} 0 & , i \in \{0,1\} \\ \frac{i}{4} & , i \in \{2,3\} \\ \frac{5}{4} & , i = 4 \end{cases}.$$

The function $\mathsf{sep}(i)$ captures the following intuition: Suppose that i bit strings are drawn at random, and consider the bit string formed by taking the coordinate-wise majority. Then, $\mathsf{sep}(i)$ is simply the expected sum of the fractional Hamming distances from the majority string to all the i original strings. We also extend the definition of $\mathsf{pre}(\cdot)$ to sets $S \subseteq \Sigma^* \times \mathbb{N}$ as follows :

$$\mathsf{pre}_i(S) = \{s_{\le i} \mid (s,l) \in S, l < i \le |s|\}.$$
$$\mathsf{pre}(S) = \bigcup_{i>0} \mathsf{pre}_i(S),$$

We are now ready to define boosted list tree codes and show that they exist.

Definition IV.5. *Let $r, L, \alpha > 0$ and $f : \Sigma^* \to \{0,1\}^r$ be a function. We say that f is an (r, L, α)-boosted list tree code if:*

- *f is an (r, L, α)-list tree code.*
- *For all $k > 0$ and all $S \subseteq \Sigma^{\le k} \times \mathbb{N}$ such that $|S| \le 4$ and $|\mathsf{pre}(S)| \ge k(1+\epsilon)$, we have:*

$$\sum_{i \in [k]} \min_{t \in \{0,1\}^r} \sum_{s \in \mathsf{pre}_i(S)} \Delta(f(s), t) \ge r(1-\epsilon) \cdot \sum_{i \in [k]} \mathsf{sep}(|\mathsf{pre}_i(S)|).$$

Theorem IV.6. *Let $\epsilon > 0$ be fixed. For all $r \ge \frac{100 \cdot \log_2(|\Sigma|+1)}{\epsilon^3}$, $L \ge \frac{100}{\epsilon^2}$, there exists an $\left(r, L, \frac{1}{2} - \epsilon\right)$-boosted list tree code.*

The proof of this theorem is deferred to the full version.

V. Binary List-Decodable Interactive Codes With Small Lists

The main result from this section is the following (see the full version for a proof):

Theorem V.1. *Let $0 < \epsilon < \frac{1}{20}$ be a parameter and $\Pi = \{T, p, \mathcal{X}^C, \mathcal{Y}^C, f^C, \mathsf{out}^C\}_{C \in \{A,B\}}$ be a protocol in the binary two party communication model as in Subsection III-A. There is a protocol $\Pi' = \{T', p', \mathcal{X}'^C, \mathcal{Y}'^C, f'^C, \mathsf{out}'^C\}_{C \in \{A,B\}}$ such that:*

1) *We have $T' = 10^{10} \cdot \frac{T}{\epsilon^{60}}$, $\mathcal{X}'^C = \mathcal{X}^C$ for all $C \in \{A,B\}$, and $\mathcal{Y}'^A = \mathcal{Y}'^B = \{0,1\}^T \times \{d \mid d : \{0,1\}^T \to \mathbb{N}\}$.*
2) *For all $x^A \in \mathcal{X}^A$, $x^B \in \mathcal{X}^B$, and all adversaries Adv' for Π', if $(L^C, \mathsf{d}^C) = \mathsf{out}'^C_{\Pi', \mathsf{Adv}'}(x^A, x^B)$ for all $C \in \{A,B\}$, then*
   a) *For all $C \in \{A,B\}$, we have $|L^C| \le 3$ and for all $s \ne s' \in L^C$, we have*
   $$\mathsf{d}^C(s) + \mathsf{d}^C(s') \ge \left(\frac{1}{4} - \frac{\epsilon}{2}\right) \cdot T'.$$
   b) *If $\mathsf{corr}_{\Pi', \mathsf{Adv}'}(x^A, x^B) \le \left(\frac{5}{32} - \epsilon\right) \cdot T'$, then $L^A \cap L^B = \{\Pi(x^A, x^B)\}$ and for all $C \in \{A,B\}$, we have:*
   $$\mathsf{d}^C(\Pi(x^A, x^B)) \le \mathsf{corr}_{\Pi', \mathsf{Adv}'}(x^A, x^B) + \frac{\epsilon T'}{2}.$$

VI. Our Simulation Procedure

We are now ready to state and prove our main result, Theorem VI.1, which is a formalization of Theorem I.1.

Theorem VI.1. *Let $0 < \epsilon < \frac{1}{20}$ be a parameter and $\Pi = \{T, p, \mathcal{X}^C, \mathcal{Y}^C, f^C, \mathsf{out}^C\}_{C \in \{A,B\}}$ be a protocol in the binary two party communication model as in Subsection III-A. There is a protocol $\hat{\Pi} = \{\hat{T}, \hat{p}, \hat{\mathcal{X}}^C, \hat{\mathcal{Y}}^C, \hat{f}^C, \widehat{\mathsf{out}}^C\}_{C \in \{A,B\}}$ such that:*

1) *We have $\hat{T} = \frac{39}{32} \cdot n_0(\epsilon/10) \cdot 10^{70} \cdot \frac{T}{\epsilon^{60}}$ where $n_0(\cdot)$ is as promised by Lemma IV.1. We also have $\hat{\mathcal{X}}^C = \mathcal{X}^C$ for all $C \in \{A,B\}$, and $\hat{\mathcal{Y}}^A = \hat{\mathcal{Y}}^B = \{0,1\}^T$.*
2) *For all $x^A \in \mathcal{X}^A$, $x^B \in \mathcal{X}^B$, and all adversaries $\hat{\mathsf{Adv}}$ for $\hat{\Pi}$, we have for all $C \in \{A,B\}$ that*
$$\mathsf{corr}_{\hat{\Pi}, \hat{\mathsf{Adv}}}(x^A, x^B) \le \left(\frac{5}{39} - \epsilon\right) \cdot \hat{T} \implies \widehat{\mathsf{out}}^C_{\hat{\Pi}, \hat{\mathsf{Adv}}}(x^A, x^B) = \Pi(x^A, x^B).$$

The rest of this paper has our proof of Theorem VI.1. Fix a parameter $0 < \epsilon < \frac{1}{20}$ and a protocol $\Pi = \{T, p, \mathcal{X}^C, \mathcal{Y}^C, f^C, \mathsf{out}^C\}_{C \in \{A,B\}}$ in the binary two party communication model. At the cost of blowing up the number of rounds in Π by a factor of 2, we can assume that Π is an alternating binary protocol, *i.e.*, Alice transmits in the odd rounds of Π and Bob transmits in the even rounds of Π. Thus, henceforth, we assume Π has $2T$ rounds and

$p_{2i-1} = A$ and $p_{2i} = B$ for all $i \in [T]$. We let Π_{List} denote the protocol promised by Theorem V.1 with the parameters $\epsilon/10$ and Π and define $\hat{N} = \frac{\hat{T}}{39}$ for convenience. Observe that Π_{List} has $32\hat{N}$ rounds.

We shall use ECC to denote the functions $\mathsf{ECC}_{4\hat{N},\epsilon/10}$ promised by Lemma IV.1 and C_n will denote the function promised by Lemma IV.2. The protocol $\hat{\Pi}$ that proves Theorem VI.1 is described in Algorithm 2 (Alice's side) and Algorithm 3 (Bob's side).

Algorithm 2 Alice's side of the simulation protocol

Input: An input $x^A \in \mathcal{X}^A$.
Output: A transcript $\pi \in \{0,1\}^{2T}$.

Phase 1:

Run Π_{List} with input x^A and get output (L, d). Interpret L as a list by ordering the elements in some canonical way.

Phase 2:

Listen for the next $4\hat{N}$ rounds. Let $\sigma \in \{0,1\}^{4\hat{N}}$ be the symbols received.

Compute Output:

For $l \in L$, set $e(l) \leftarrow \min_{L' : |L'| \le 3 \text{ and } L \cap L' = \{l\}} \Delta\left(\mathsf{ECC}(L'), \sigma\right)$ and let $E(l)$ be the minimizer.
$\pi \leftarrow \arg\min_{l \in L} \mathsf{d}(l) + e(l)$.
Set $\mathsf{p} \in [3]$ to be the position of π in $E(\pi)$.

Phase 3:

For the next $3\hat{N} - \frac{3}{2} \cdot e(\pi)$ rounds, transmit $\mathsf{C}_{\hat{N} - \frac{e(\pi)}{2}}(\mathsf{p})$.

For the remaining $\frac{3}{2} \cdot e(\pi)$ rounds, transmit $\mathsf{C}_{\frac{e(\pi)}{2}}(0)$.

Algorithm 3 Bob's side of the simulation protocol

Input: An input $x^B \in \mathcal{X}^B$.
Output: A transcript $\pi \in \{0,1\}^{2T}$.

Phase 1:

Run Π_{List} with input x^B and get output (L, d). Interpret L as a list by ordering the elements in some canonical way.

Phase 2:

For the next $4\hat{N}$ rounds, transmit $\mathsf{ECC}(L) \in \{0,1\}^{4\hat{N}}$, symbol by symbol.

Phase 3:

Listen in the remaining $3\hat{N}$ rounds. Let $\tau \in \{0,1\}^{3\hat{N}}$ be the symbol received.

Compute Output:

For $\tilde{\mathsf{p}} \in \{0,1,2,3\}$, set $e(\tilde{\mathsf{p}}) \leftarrow \Delta\left(\mathsf{C}_{\hat{N}}(\tilde{\mathsf{p}}), \tau\right)$.

$$\mathsf{q} \leftarrow \arg\min_{j \in [3]} \Big(\mathsf{d}(L_j) + \min\Big(e(j), e(0) + 3\hat{N} \cdot \mathbb{1}\Big(j \neq \arg\min_{j' \in [3]} \mathsf{d}(L_{j'})\Big)\Big)\Big).$$

$\pi \leftarrow L_{\mathsf{q}}$.

A. *Proof of Theorem VI.1*

We now prove Theorem VI.1.

Proof of Theorem VI.1: Observe that item 1 of Theorem VI.1 follows straightforwardly from our definition of $\hat{\Pi}$ in Algorithm 2 and Algorithm 3.

For item 2 of Theorem VI.1, we fix inputs x^A and x^B for Alice and Bob respectively and an adversary $\hat{\mathsf{Adv}}$ for the protocol $\hat{\Pi}$. As our protocol is deterministic, fixing x^A, x^B, and $\hat{\mathsf{Adv}}$ fixes the values of all the variables in Algorithm 2 and Algorithm 3. In our analysis, we shall use var^A (respectively, var^B) to denote the value of variable var in Algorithm 2 (resp. Algorithm 3). We shall drop the superscript if the variable appears in only one of Algorithm 2 and Algorithm 3. Observe that, with this notation, we have for all $C \in \{A, B\}$ that $\widehat{\mathsf{out}}^C_{\hat{\Pi},\hat{\mathsf{Adv}}}(x^A, x^B) = \pi^C$ and in order to show item 2 of Theorem VI.1, we have to show for all $C \in \{A, B\}$ that

$$\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}}}(x^A, x^B) \le \left(\frac{5}{39} - \epsilon\right) \cdot \hat{T} \implies \pi^C = \Pi(x^A, x^B).$$

To start with, we use item 2 of Theorem V.1 to get that $\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}}}(x^A, x^B) \le \left(\frac{5}{39} - \epsilon\right) \cdot \hat{T}$ implies $L^A \cap L^B = \{\Pi(x^A, x^B)\}$ and for all $C \in \{A, B\}$ and $s, s' \in L^C$, we have:

$$\mathsf{d}^C(\Pi(x^A, x^B)) \le \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) + 2\epsilon\hat{N}. \quad (3)$$

$$\mathsf{d}^C(s) + \mathsf{d}^C(s') \ge \left(\frac{1}{4} - \frac{\epsilon}{20}\right) \cdot 32\hat{N}. \quad (4)$$

We divide the rest of this proof into two parts and use the fact that $L^A \cap L^B = \{\Pi(x^A, x^B)\}$ implicitly throughout.

- **Showing $\pi^A = \Pi(x^A, x^B)$:** To start, observe that $e^A(\Pi(x^A, x^B)) \le \mathsf{corr}_{\Pi',\mathsf{Adv}',(32\hat{N},39\hat{N}]}(x^A, x^B)$. Now, if $\pi^A \neq \Pi(x^A, x^B)$, using the fact that $\pi^A \in L^A$ by definition, we get:

$$\begin{aligned}
&2 \cdot \left(\mathsf{d}^A(\Pi(x^A, x^B)) + e^A(\Pi(x^A, x^B))\right) \\
&\le 2 \cdot \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}}}(x^A, x^B) && \text{(Equation 3)} \\
&< (10 - 5\epsilon)\hat{N} \\
&\le \mathsf{d}^A(\Pi(x^A, x^B)) + \mathsf{d}^A(\pi^A) \\
&\qquad + \Delta\left(\mathsf{ECC}(E(\Pi(x^A, x^B))), \mathsf{ECC}(E(\pi^A))\right) \\
&\qquad && \text{(Equation 4 and Lemma IV.1)}
\end{aligned}$$

$$\leq \mathsf{d}^A(\Pi(x^A, x^B)) + \mathsf{d}^A(\pi^A) + e^A(\Pi(x^A, x^B)) + e^A(\pi^A),$$
(Triangle inequality)

a contradiction to the choice of π^A.

- **Showing $\boldsymbol{\pi^B = \Pi(x^A, x^B)}$:**
  We further break this part of the proof into two cases:
  - **When $\boldsymbol{E(\pi^A) = L^B}$:** In this case, we get that $L^B_{\mathsf{p}} = \pi^A = \Pi(x^A, x^B)$. Now, if $\pi^B \neq \Pi(x^A, x^B)$ then $\mathsf{q} \neq \mathsf{p}$ and we get:

$$\begin{aligned}
&2 \cdot \left(d^B(L^B_{\mathsf{p}}) + e^B(\mathsf{p})\right) \\
&\leq 2 \cdot \left(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) + e^B(\mathsf{p})\right) \\
&\leq 2 \cdot \left(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) + \Delta\left(\mathsf{C}_{\hat{N}}(\mathsf{p}), \tau\right)\right) && \text{(Definition of } e^B(\cdot)\text{)} \\
&\leq 2 \cdot \Big(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) \\
&\qquad + \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(36\hat{N}:39\hat{N}]}(x^A, x^B) \\
&\qquad + e^A(\Pi(x^A, x^B))\Big) && \text{(Line 12 and Line 13)} \\
&< (10 - 5\epsilon)\hat{N} \\
&\quad (e^A(\Pi(x^A, x^B)) \leq \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(32\hat{N}:36\hat{N}]}(x^A, x^B)) \\
&\leq \min\left(\Delta\left(\mathsf{C}_{\hat{N}}(\mathsf{p}), \mathsf{C}_{\hat{N}}(\mathsf{q})\right), \Delta\left(\mathsf{C}_{\hat{N}}(\mathsf{p}), \mathsf{C}_{\hat{N}}(0)\right)\right) \\
&\qquad + \mathsf{d}^B(L^B_{\mathsf{p}}) + \mathsf{d}^B(L^B_{\mathsf{q}}) && \text{(Equation 4 and Lemma IV.2)} \\
&\leq \mathsf{d}^B(L^B_{\mathsf{p}}) + \mathsf{d}^B(L^B_{\mathsf{q}}) \\
&\qquad + e^A(\mathsf{p}) + \min\left(e^B(\mathsf{q}), e^B(0)\right), && \text{(Triangle inequality)}
\end{aligned}$$

    a contradiction to the choice of q.
  - **When $\boldsymbol{E(\pi^A) \neq L^B}$:** In this case, we first show that $\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(32\hat{N}:36\hat{N}]}(x^A, x^B) \geq (1-\epsilon)\hat{N}$. We have:

$$\begin{aligned}
&\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(32\hat{N}:36\hat{N}]}(x^A, x^B) \\
&\geq \Delta\left(\mathsf{ECC}(L^B), \sigma\right) \\
&\geq \max\Big(\Delta\left(\mathsf{ECC}(E(\pi^A), \sigma\right) \\
&\qquad , (2-\epsilon)\hat{N} - \Delta\left(\mathsf{ECC}(E(\pi^A), \sigma\right)\Big) \\
&\qquad \text{(Definition of } E \text{ and Lemma IV.1)} \\
&\geq (1-\epsilon)\hat{N}.
\end{aligned}$$

    Therefore, we have $\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) < (4-\epsilon)\hat{N}$. This implies that $\Pi(x^A, x^B) = L^B_{\mathsf{q}^*}$ where $\mathsf{q}^* = \arg\min_{j' \in [3]} \mathsf{d}^B(L_{j'})$. If not, then, as $\Pi(x^A, x^B) \in L^B$, we have by Theorem V.1,

$$\begin{aligned}
2 \cdot \mathsf{d}^B(\Pi(x^A, x^B)) &\leq 2 \cdot \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) \\
&< (8 - 2\epsilon)\hat{N} \\
&\leq \mathsf{d}^B(\Pi(x^A, x^B)) + \mathsf{d}^B(L^B_{\mathsf{q}^*}), && \text{(Equation 4)}
\end{aligned}$$

    a contradiction to the choice of q^*. Now that we have $\Pi(x^A, x^B) = L^B_{\mathsf{q}^*}$, if $\pi^B \neq \Pi(x^A, x^B)$, then $\mathsf{q} \neq \mathsf{q}^*$ and we have

$$\begin{aligned}
&2 \cdot \left(\mathsf{d}^B(L^B_{\mathsf{q}^*}) + e^B(0)\right) \\
&\leq 2 \cdot \left(\mathsf{corr}_{\Pi',\mathsf{Adv}',[32\hat{N}]}(x^A, x^B) + e^B(0)\right) \\
&\leq 2 \cdot \left(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) + \Delta\left(\mathsf{C}_{\hat{N}}(0), \tau\right)\right) && \text{(Definition of } e^B(\cdot)\text{)} \\
&\leq 2 \cdot \Big(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) \\
&\qquad + \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(36\hat{N}:39\hat{N}]}(x^A, x^B) \\
&\qquad + 2\hat{N} - e^A(\Pi(x^A, x^B))\Big) && \text{(Line 12 and Line 13)} \\
&\leq 2 \cdot \Big(\mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},[32\hat{N}]}(x^A, x^B) \\
&\qquad + \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(36\hat{N}:39\hat{N}]}(x^A, x^B) \\
&\qquad + \Delta\left(\mathsf{ECC}(L^B), \sigma\right) + 2\epsilon\hat{N}\Big) && (E(\pi^A) \neq L^B) \\
&< (10 - 5\epsilon)\hat{N} \\
&\quad (e^A(\Pi(x^A, x^B)) \leq \mathsf{corr}_{\hat{\Pi},\hat{\mathsf{Adv}},(32\hat{N}:36\hat{N}]}(x^A, x^B)) \\
&\leq \mathsf{d}^B(L^B_{\mathsf{q}^*}) + d^B(L^B_{\mathsf{q}}) + \Delta\left(\mathsf{C}_{\hat{N}}(\mathsf{q}), \mathsf{C}_{\hat{N}}(0)\right) \\
&\qquad \text{(Equation 4 and Lemma IV.2)} \\
&\leq \mathsf{d}^B(L^B_{\mathsf{q}^*}) + d^B(L^B_{\mathsf{q}}) + e^B(0) + e^B(\mathsf{q}), && \text{(Triangle inequality)}
\end{aligned}$$

    a contradiction to the choice of q.

■

References

[1] L. J. Schulman, "Coding for interactive communication," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1745–1756, 1996. 1, 2, 4

[2] M. Braverman and A. Rao, "Towards coding for maximum errors in interactive communication," in *Symposium on Theory of computing (STOC)*. ACM, 2011, pp. 159–166. 1, 2, 3, 4, 5, 6

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

[4] M. Braverman and K. Efremenko, "List and unique coding for interactive communication in the presence of adversarial noise," *SIAM Journal on Computing*, vol. 46, no. 1, pp. 388–428, 2017. [Online]. Available: https://doi.org/10.1137/141002001 2, 3, 4, 6

[5] K. Efremenko, R. Gelles, and B. Haeupler, "Maximal noise in interactive communication over erasure channels and channels with feedback," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4575–4588, 2016. 2, 3

[6] V. M. Blinovskii, "Bounds for codes in the case of finite-volume list decoding," *Problemy Peredachi Informatsii*, vol. 22, no. 1, pp. 11–25, 1986. 2, 6

[7] N. Alon, B. Bukh, and Y. Polyanskiy, "List-decodable zero-rate codes," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1657–1667, 2018. 2

[8] L. J. Schulman, "Communication on noisy channels: A coding theorem for computation," in *Foundations of Computer Science (FOCS)*. IEEE, 1992, pp. 724–733. 2

[9] ——, "Deterministic coding for interactive communication," in *Symposium on Theory of computing (STOC)*. ACM, 1993, pp. 747–756. 2

[10] R. Gelles, A. Moitra, and A. Sahai, "Efficient and explicit coding for interactive communication," in *Foundations of Computer Science (FOCS)*. IEEE, 2011, pp. 768–777. 2

[11] M. Braverman, "Towards deterministic tree code constructions," in *Innovations in Theoretical Computer Science (ITCS)*. ACM, 2012, pp. 161–167. 2

[12] G. Kol and R. Raz, "Interactive channel capacity," in *Symposium on Theory of computing (STOC)*, 2013, pp. 715–724. 2

[13] Z. Brakerski, Y. T. Kalai, and M. Naor, "Fast interactive coding against adversarial noise," *Journal of the ACM (JACM)*, vol. 61, no. 6, p. 35, 2014. 2

[14] R. Gelles, A. Moitra, and A. Sahai, "Efficient coding for interactive communication," *IEEE Transactions on Information Theory*, vol. 60, no. 3, pp. 1899–1913, 2014. 2

[15] R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson, "Explicit capacity approaching coding for interactive communication," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6546–6560, 2018. [Online]. Available: https://doi.org/10.1109/TIT.2018.2829764 2

[16] M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky, "Coding for interactive communication correcting insertions and deletions," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6256–6270, 2017. 2

[17] M. Ghaffari, B. Haeupler, and M. Sudan, "Optimal error rates for interactive coding i: Adaptivity and other settings," in *Symposium on Theory of computing (STOC)*, 2014, pp. 794–803. 2, 3

[18] S. Agrawal, R. Gelles, and A. Sahai, "Adaptive protocols for interactive communication," in *Information Theory (ISIT)*. IEEE, 2016, pp. 595–599. 2

[19] K. Efremenko, G. Kol, and R. Saxena, "Interactive error resilience beyond 2/7," in *Symposium on Theory of Computing (STOC)*. ACM, 2020. 2, 3

[20] ——, "Optimal error resilience of adaptive message exchange," 2020, manuscript. 2, 3

[21] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman, "Optimal coding for streaming authentication and interactive communication," *IEEE Transactions on Information Theory*, vol. 61, no. 1, pp. 133–145, 2015. 3

[22] A. A. Sherstov and P. Wu, "Optimal interactive coding for insertions, deletions, and substitutions," in *Foundations of Computer Science (FOCS)*, 2017, pp. 240–251. 3

[23] B. Haeupler, A. Shahrasbi, and E. Vitercik, "Synchronization strings: Channel simulations and interactive coding for insertions and deletions," in *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018, pp. 75:1–75:14. 3

[24] M. Ghaffari and B. Haeupler, "Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding," in *Foundations of Computer Science (FOCS)*, ser. FOCS, 2014, pp. 394–403. 3, 6

[25] O. Shayevitz, "On error correction with feedback under list decoding," in *IEEE International Symposium on Information Theory*. IEEE, 2009, pp. 1253–1257. 3

APPENDIX

We now sketch the proof of a claim made in Subsection II-B that a list decodable protocol with a binary alphabet for the message exchange task requires exponential sized lists if the fraction of errors is larger than $\frac{1}{4}$.

Proof sketch: Let $\epsilon > 0$ be fixed and let Π be a non-adaptive list decodable protocol for the message exchange task that is resilient to $\frac{1}{4}+\epsilon$ fraction of errors. Without loss of generality, let Alice be the party that speaks less in Π and note that this is well defined as Π is non-adaptive.

Fix Bob's input x^B arbitrarily. For every input x^A for Alice, define the set $S(x^A)$ of Bob's transcripts as follows: We say that a transcript $\pi \in S(x^A)$ iff there exists an adversary Adv for Π that corrupts at most $\frac{1}{4}+\epsilon$ fraction of messages and ensures that the transcript received by Bob is π when Alice and Bob have inputs x^A and x^B respectively.

Also define, for a transcript π received by Bob, the list $L(\pi)$ to be the list output by Bob when he receives this transcript and his input is x^B. Observe that, for all $\pi \in S(x^A)$, we must have $x^A \in L(\pi)$. We get that

$$\sum_{x^A} |S(x^A)| \leq \sum_{\pi} |L(\pi)| \leq |\Pi_B| \cdot \max_{\pi} |L(\pi)|,$$

where Π_B is the set of all transcript that Bob can potentially receive. Next, we observe that, for all x^A, a random transcript $\pi \in S(x^A)$ with probability at least $\frac{1}{2}$. If $\mathcal{X}^A$ denote the set of all Alice's inputs, we get that:

$$\frac{1}{2}\cdot|\Pi_B|\cdot|\mathcal{X}^A| \leq |\Pi_B|\cdot\max_{\pi}|L(\pi)| \implies \max_{\pi}|L(\pi)| \geq \frac{1}{2}\cdot|\mathcal{X}^A|.$$

The proof is done with the observation that $\max_\pi |L(\pi)|$ is the list size of the protocol. ∎