The Optimal Error Resilience of Interactive Communication Over Binary Channels

Meghal Gupta[*]
Microsoft Research

Rachel Yun Zhang[†]
Massachusetts Institute of Technology

November 7, 2021

Abstract

In interactive coding, Alice and Bob wish to compute some function f of their individual private inputs x and y. They do this by engaging in a non-adaptive (fixed order, fixed length) interactive protocol to jointly compute $f(x, y)$. The goal is to do this in an error-resilient way, such that even given some fraction of adversarial corruptions to the protocol, both parties still learn $f(x, y)$.

In this work, we study the optimal error resilience of such a protocol in the face of adversarial bit flip or erasures. While the optimal error resilience of such a protocol over a large alphabet is well understood, the situation over the *binary* alphabet has remained open. In this work, we resolve this problem of determining the optimal error resilience over binary channels. In particular, we construct protocols achieving $\frac{1}{6}$ error resilience over the binary bit flip channel and $\frac{1}{2}$ error resilience over the binary erasure channel, for both of which matching upper bounds are known. We remark that the communication complexity of our binary bit flip protocol is polynomial in the size of the inputs, and the communication complexity of our binary erasure protocol is linear in the size of the minimal noiseless protocol computing f.

[*]Email: `meghal@mit.edu`
[†]Email: `rachelyz@mit.edu`

Contents

1 Introduction

Interactive coding is an interactive analogue of error correcting codes [Sha48, Ham50], that was introduced in the seminal work of Schulman [Sch92, Sch93, Sch96] and has been an active area of study since. While error correcting codes address the problem of sending a *message* in a way that is resilient to error, interactive coding addresses the problem of converting an *interactive protocol* to an error resilient one.

Suppose two parties, Alice and Bob, each with a private input, engage in a protocol π_0 to jointly compute a function f of their private inputs. *Given such a protocol π_0, can we design a protocol π that computes f and at the same time is resilient to adversarial errors?* Schulman [Sch96] answered the question in the affirmative, presenting a scheme that is resilient to $\frac{1}{240}$[1] adversarial corruptions (bit flips) over a binary channel with a constant information rate.[2] This work begs the natural question: *what is the maximum error resilience possible?* This is precisely the focus of our work.

Two natural types of corruption to consider are *bit flip* (where the adversary can replace a symbol with one of their choice) and *erasure* (where the adversary can replace a symbol with $\perp$). In both of these settings, there are known protocols that achieve optimal error resilience for large constant-sized alphabets. In the bit flip setting, Braverman and Rao [BR11] constructed a protocol which achieves the optimal error resilience of $\frac{1}{4}$. In the erasure setting, [FGOS15, EGH16] constructed protocols achieving the optimal error resilience $\frac{1}{2}$. Corresponding impossibility bounds are known [BR11, FGOS15].

However, despite much effort, the optimal error resilience for both types of corruption over a *binary* alphabet is still unknown. For both types of corruption, a protocol achieving error resilience of r over a large alphabet trivially translates to a protocol over a binary alphabet with error resilience of $\frac{r}{2}$ by replacing every letter of the large alphabet with a binary error correcting code of relative distance $\frac{1}{2}$. Thus, the results of [BR11, FGOS15] give protocols that achieve error resilience $\frac{1}{8}$ over the binary bit flip channel and $\frac{1}{4}$ over the binary erasure channel.

Unfortunately, the corresponding impossibility bounds [BR11, FGOS15] for large alphabets do not lead to matching impossibility bounds in the binary case. Over the binary bit flip channel, the best known impossibility bound is $\frac{1}{6}$ [EGH16], and over the binary erasure channel, the best known impossibility bound is $\frac{1}{2}$ [FGOS15]. Pinning down the exact constant between $\frac{1}{8}$ and $\frac{1}{6}$, and between $\frac{1}{4}$ and $\frac{1}{2}$, has been an intriguing open problem.

There has been some recent work towards this goal. Over the binary bit flip channel, [EKS20] broke the $\frac{1}{8}$ barrier for the first time, describing a protocol that achieves $\frac{5}{39}$ resilience to adversarial bit flips. Over the binary erasure channel, [EGH16] gives a protocol achieving a $\frac{1}{3}$ resilience to erasures. Nonetheless, the exact values of the optimal error resilience over the binary bit flip and erasure channels have remained unknown since their initial active investigation by [BR11] in 2011 and [FGOS15] in 2013.

In this work, we resolve the question of the optimal error resilience of a non-adaptive (fixed order, fixed length) protocol over a binary alphabet. Specifically, we show that the known impossibility bounds are tight: *we construct protocols achieving error resilience $\frac{1}{6}$ to adversarial bit flips and $\frac{1}{2}$ to adversarial erasures.*

[1]Whenever we say that a protocol has resilience $r \in [0, 1]$ in the introduction and overview, we mean that for any ϵ, there exists an instantiation that achieves resilience $r - \epsilon$.

[2]Constant information rate means that the error-resilient protocol incurs only a constant multiplicative overhead to the communication complexity.

1.1 Our Results

We show the following result for two-party communication over the binary bit flip channel.

Theorem 1.1. *For any function f, there exists a non-adaptive binary interactive protocol computing the function $f(x, y)$ that is resilient to $\frac{1}{6} - \epsilon$ adversarial bit flips with probability $1 - 2\exp(-\Omega(\epsilon n))$. For inputs of size n, the communication complexity is $O_\epsilon(n^2)$ and the runtime of the parties is $C(\epsilon) \cdot n^{O(1)}$ for some constant $C(\epsilon)$.*

Until our result, it was only known how to achieve $\frac{1}{6}$ error resilience if Alice and Bob are given the extra power to know, instantly, what the other party received at the other end of the channel when they send a message. This additional power is known as *feedback* [EGH16, GH17, Ber64, Ber68, Zig76, SW92] and is given at no cost. It is thought to give considerably more power to the parties, as there is never any uncertainty about what the other party has heard so far. An error resilience of $\frac{1}{6}$ is known to be tight even in this model [EGH16], but all protocol constructions rely crucially on the fact that the sender can always send specifically the piece of information about their input that the receiver needs to hear.

Our $\frac{1}{6}$ error resilient protocol shows that protocols *without feedback* can do just as well. The surprising implication is that the ability to know what messages are received by the other party actually grants no additional power!

Remark 1.2. The communication complexity and runtime of our protocol are polynomial in the size of Alice and Bob's inputs, rather than on the minimal size of an error-free protocol π_0 computing f, which can be exponentially smaller. This may result in an exponential blowup in communication complexity relative to a corresponding error-free protocol π_0 for simpler functions Alice and Bob might want to compute. We leave it as an open problem to construct (or disprove the existence of) a protocol with $\left(\frac{1}{6} - \epsilon\right)$-error resilience whose communication complexity and runtime are polynomial in $|\pi_0|$ or ideally even linear.

Over the binary erasure channel, we show the following efficient result:

Theorem 1.3. *For any interactive binary protocol π_0 computing a function $f(x, y)$ of Alice and Bob's inputs $x, y \in \{0, 1\}^n$, there exists a non-adaptive interactive binary protocol π computing $f(x, y)$ that is resilient to $\frac{1}{2} - \epsilon$ adversarial erasures. The communication complexity and runtime for each party are both $O_\epsilon(|\pi_0|)$.*

We note that $\frac{1}{2}$ is in fact the maximal possible erasure resilience of a two-party interactive protocol over *any* alphabet, as the adversary can simply erase all messages of the party that speaks less. Previous work [FGOS15, EGH16] constructed protocols resilient to $\frac{1}{2}$ erasures over larger alphabets. In our work, we show that the alphabet size makes no difference to the optimal erasure resilience. This contrasts with the bit flip model, where an error resilience of $\frac{1}{4}$ is attainable over large alphabets, while over the binary alphabet it is capped at $\frac{1}{6}$.

2 Technical Overview

We now give an overview of our two protocols achieving maximal error resilience over the binary bit flip channel and binary erasure channel respectively.

2.1 Our Binary Protocol Resilient to $\frac{1}{6}$ Bit Flips

We begin with the following (flawed) approach, which achieves an error resilience of $\frac{1}{8}$: Alice sends an error correcting code $\mathsf{ECC}(x)$, and Bob sends $\mathsf{ECC}(y)$. Since ECC has relative distance of at most $\frac{1}{2}$, the adversary can simply flip $\frac{1}{4}$ of the bits of the party that speaks less so that the other party cannot distinguish between two inputs.

Recall that the maximum error resilience of any binary two-party protocol is at most $\frac{1}{6}$ of the total communication, or $\frac{1}{3}$ of either party's communication. The above flawed protocol only had error resilience of $\frac{1}{4}$ of either party's communication. In order to increase this to $\frac{1}{3}$ for one of the parties, we introduce a new *question-answer* approach: in each round of interaction, Alice asks a question (encoded with an ECC) about Bob's input, and then Bob responds with one of *four* answers. More specifically,

- Alice tracks a guess $\hat{y}$ for Bob's input y initially set to $\emptyset$, and a counter c_A indicating her confidence for $\hat{y}$ initially set to 0. Each round, she sends $\hat{y}$ encoded in an ECC to Bob as her question.

- Bob responds with one of four operations to do to $\hat{y}$ as his answer: append 0 (0), append 1 (1), delete the last bit ($\leftarrow$), or "bingo – you got it right!" ($\bullet$).

- Alice updates based on the answer as follows: if she receives $\bullet$, she increases c_A by 1 since Bob is informing her that her guess is correct. If she receives 0, 1, or $\leftarrow$ and if $c_A = 0$, she makes the corresponding adjustment to the string $\hat{y}$ (append 0 or 1, or delete the last bit). Otherwise if $c_A \neq 0$, she simply decreases c_A by 1 without making the corresponding adjustment to $\hat{y}$.

Ultimately, as long as Alice receives Bob's correct answer at least $|y|$ more times than she receives a wrong answer, she will output the correct answer. The key point is that these four responses from Bob can have distance $\frac{2}{3}$ (e.g., 000, 110, 011, 101). Now, if the adversary simply corrupts Bob's answers, she'd have to corrupt $\approx \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$ of Bob's rounds.

While this achieves $\frac{1}{3}$ resilience for Bob's communication, Alice's communication is still only $\frac{1}{4}$-error resilient as she is sending an error correcting code. In order to attain $\frac{1}{3}$ error resilience for *both* parties, both parties will need to *simultaneously ask and answer a question* each round. Specifically,

- Alice and Bob each track a guess $\hat{y}$ or $\hat{x}$ respectively for the other party's input, as well as a counter c_A or c_B.

- Each round, Alice sends $\mathsf{ECC}(\hat{y}, x^*, \delta)$: $\hat{y}$ is her question, x^* is the question she just heard from Bob and δ is the instruction that brings x^* one character closer to her input x (or $\bullet$ if $x^* = x$). Similarly, Bob sends $\mathsf{ECC}(\hat{x}, y^*, \delta)$.

- When Alice receives the message $\mathsf{ECC}(x^*, y^*, \delta^*)$[3] from Bob, she updates $\hat{y}$ according to the instruction δ^*, but only if $y^* = \hat{y}$ (intuitively, because she should not update if Bob is answering the wrong question). Bob does the same.

[3] Alice may also receive a message that is only partially corrupted (i.e., does not correspond to a codeword of the form $\mathsf{ECC}(x^*, y^*, \delta^*)$). We address partial corruptions later, and for now assume that the adversary must always corrupt a message to another valid codeword.

The ECC we use has relative distance $\geq \frac{1}{2}$ between all pairs of codewords, and $\geq \frac{2}{3}$ for pairs of codewords of the form $\mathsf{ECC}(x', y', \delta_0)$ and $\mathsf{ECC}(x', y', \delta_1)$ with $\delta_0 \neq \delta_1$ (or equivalently $\mathsf{ECC}(y', x', \delta_0)$ and $\mathsf{ECC}(y', x', \delta_1)$ with $\delta_0 \neq \delta_1$). (We construct such an ECC explicitly in Claim 4.5.) Now if the adversary only corrupts Bob's answer, specifically only corrupting δ, it will require $\approx \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$ corruptions of Bob's communication for Alice to output the wrong y.

However, it seems as if we have lost the advantage we gained earlier: the adversary can corrupt both Bob's answer δ *and* his question $\hat{x}$ in half the rounds so that Alice receives $\mathsf{ECC}(x', \hat{y}, \delta')$, where $x' \neq \hat{x}$ and $\delta' \neq \delta$. This attack every other round only requires corrupting $\frac{1}{2}$ rather than $\frac{2}{3}$ of Bob's message had they only corrupted δ and not $\hat{x}$ as well. Alice still does not make progress over time, and the adversary only needs to corrupt $\approx \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ of Bob's total communication.

Let us analyze this situation more closely. We count progress as the number of good updates (getting the guess $\hat{y}$ or $\hat{x}$ closer to the other party's input or adjusting c_A or c_B correctly) minus the number of bad updates (getting $\hat{y}$ or $\hat{x}$ further from the other party's input or adjusting c_A or c_B incorrectly). The adversary can corrupt $\frac{1}{2}$ of the bits in Bob's message to get -1 progress for Alice and 0 progress for Bob (since $x' \neq \hat{x}$, Alice answers the wrong question for Bob so that he performs no update), from an original progress count of $(+1, +1)$ had no corruptions occurred. So, when the adversary corrupts half of Bob's messages, Alice's final progress is 0, so that she does not know y at the end of the protocol. However, Bob's progress still increases at a steady rate! If there were a way to exploit this, so that when Bob has made a lot of progress (i.e., $\hat{x} = x$) it becomes harder to cause negative progress for Alice, perhaps we could achieve our goal of $\frac{1}{3}$ error resilience for each party.

To do this, we make the following probabilistic change to the way Alice (likewise Bob) makes updates to her current guess $\hat{y}$ upon receiving $\mathsf{ECC}(x^*, \hat{y}, \delta^*)$:

- Alice only updates $\hat{y}$ with probability 1 if the question she just received is equal to her input (i.e., $x^* = x$), otherwise (if $x^* \neq x$) she updates $\hat{y}$ with probability only 0.5.

This way, when Bob has made lots of progress so that $\hat{x} = x$, if the adversary corrupts Bob's message to be $\mathsf{ECC}(x^* \neq x, \hat{y}, \delta^*)$, Alice only updates $\hat{y}$ with probability 0.5 — this is -0.5 progress instead of -1! Now, the adversary has to corrupt two out of every three of Bob's messages in order for Alice to remain at 0 progress, bringing us up to a $\frac{1}{3}$ corruption rate for Bob's messages. The key idea is that corrupting both Bob's question $\hat{x}$ and instruction δ, which only involves corrupting $\frac{1}{2}$ of the message rather than $\frac{2}{3}$, becomes less damaging to Alice's progress than corrupting only δ, which requires $\frac{2}{3}$ corruption. This only works when Bob knows x (so $\hat{x} = x$), but this is exactly what we wanted — we needed to prevent the adversary from being able to cheaply keep Alice from making progress when Bob has made a lot of progress.

This change in update probability introduces a new situation, which is that when both Alice and Bob have made *little* progress ($\hat{y} \neq y$ and $\hat{x} \neq x$), the adversary can corrupt Bob's message to be $\mathsf{ECC}(x \neq \hat{x}, \hat{y}, \delta')$ so that Alice performs a bad update with probability 1, i.e. $(-1, 0)$ progress, from an original $(+0.5, +0.5)$ progress without corruption. Then, if the adversary corrupts one message every two rounds of interaction, the total progress between Alice and Bob remains 0! To remedy this, we have Bob (and similarly Alice) perform an additional update:

- When Bob receives a message of the form $\mathsf{ECC}(y^*, x^* \neq \hat{x}, \bullet)$, he brings his current guess $\hat{x}$ one character closer to x^* (or adjusts c_B by 1) with probability 0.5.

Now, when Bob receives Alice's response $\mathsf{ECC}(\hat{y}, x, \bullet)$ to his corrupted question, he performs a $+0.5$ update, so that the total effect of the adversary's corruption to Bob's message is $(-1, +0.5)$, i.e.

-0.5 collective progress. This makes it so that when $\hat{y} \neq y$ and $\hat{x} \neq x$, the adversary must corrupt on average $\frac{1}{3}$ of a party's messages, or $\frac{1}{6}$ the total communication, in order to prevent collective progress from being made. (Then, when a lot of collective progress has been made, at least one of the parties, say Bob, must have $\hat{x} = x$, so as discussed above the adversary must corrupt at least $\frac{1}{6}$ of the communication to prevent Alice from also making progress.)

Our protocol. To summarize, here is an outline of our protocol from Alice's perspective, assuming that she always receives a full codeword. She holds a guess $\hat{y}$ and a confidence counter c_A, initialized to $\emptyset$ and 0 respectively. Each round, she does the following:

- Alice receives $\mathsf{ECC}(x^*, y^*, \delta^*)$ from Bob.
- She updates $\hat{y}$ and c_A as follows:
  - If $y^* = \hat{y}$ and $x^* = x$, she updates $(\hat{y}, c_A)$ according to δ^* with probability 1.
  - If $y^* = \hat{y}$ and $x^* \neq x$, she updates $(\hat{y}, c_A)$ according to δ^* with probability 0.5.
  - If none of the first two conditions hold, meaning that $y^* \neq \hat{y}$, and if $\delta^* = \bullet$, she does the following update with probability 0.5: she decreases c_A by 1 if $c_A > 0$ and otherwise brings $\hat{y}$ one step closer to y^*.
- Finally, she computes δ which brings x^* closer to x (or $\bullet$ if $x^* = x$) and sends Bob $\mathsf{ECC}(\hat{y}, x^*, \delta)$.

Dealing with partial corruptions. It turns out this protocol works as long as Alice and Bob always receive a (possibly incorrect) codeword. We are almost done, but we need to specify their behaviors when they receive partially corrupted messages. To do this, we say that when Alice receives a message from Bob, she "rounds" to the nearest full codeword and does the corresponding update with some lower probability depending on the distance to the codeword. Precisely:

- Alice rounds to a codeword $\mathsf{ECC}(x, \hat{y}, \delta)$ if the received message is relative distance $d^* < \frac{1}{3}$ away, and otherwise she rounds to a codeword $\mathsf{ECC}(x^*, y^*, \delta^*)$ if the relative distance d^* is $< \frac{1}{6}$. (At most one such rounded codeword can exist since the relative distance between any two codewords is $\geq \frac{1}{2}$ and between two codewords $\mathsf{ECC}(x, \hat{y}, \delta_0)$ and $\mathsf{ECC}(x, \hat{y}, \delta_1)$ it is $\frac{2}{3}$.)
- She performs the corresponding update with probability $1 - 3d^*$ or $0.5 - 3d^*$ respectively and then replies with $\hat{y}$, the value of Bob's question x or x^* in the rounded codeword, and an instruction δ on how to update it, all jointly encoded with ECC. If no rounded codeword exists, she does no update and sends a message of the form $\mathsf{ECC}(\hat{y}, x, \bullet)$.

For a formal description of the protocol and a detailed analysis, we refer the reader to Section 4.

2.2 Our Binary Protocol Resilient to $\frac{1}{2}$ Erasures

We construct a protocol that achieves an $\frac{1}{2} - \epsilon$ erasure resilience for any $\epsilon > 0$. In our construction, we leverage a key property of the erasure channel: *communication can be delayed but not wrong.* To recreate the transcript of a (w.l.o.g. alternating) noiseless protocol π_0, Alice and Bob alternate sending bits according to the following high level strategy:

- A party (the speaker) sends their next bit of π_0 *until they are sure* that the other party (the listener) has received their message. The listener gives feedback to the speaker by sending 0 whenever the speaker's message was erased and 1 whenever they receive it.

- The speaker signals the parties to switch roles: the speaker becomes the listener and the listener becomes the speaker.

Because the adversary can only erase $\frac{1}{2} - \epsilon$ of the total messages, there must be many pairs of rounds where *both parties hear each other.* In each of these pairs of rounds, the speaker receives confirmation (in the form of a 1) that the listener received the most recent message they sent. Then the two parties unerroneously make progress in their simulation of π_0.

Intuitively, the speaker has *three* things to communicate: the bit 0, the bit 1, and switching roles. The challenge is to do this by sending only two bits.

We can let the bits 0 and 1 mean *more than two things* if we let the two bits mean different things when heard at different parts of the protocol. More specifically, we partition the protocol into $\approx \frac{n}{\epsilon}$ *blocks* of $\approx \frac{1}{\epsilon}$ rounds. In each block, the first bit that the listener receives (1 or 0) tells them whether the speaker is trying to send a bit or switch roles. In the case that the speaker is trying to send a bit, the bits received during the rest of the block tell the listener what the speaker's bit b was.

- In each block, if the speaker wants to send a bit b to add to the transcript, they send 1's until they receive signal that the listener has heard it, at which point they send b for the rest of the block.

- When they hear that the listener received the message b, they change strategies in the next block to signal the role switch: they send only 0's. For distinction, this version of speaker mode is called the passer mode.

- When the listener hears this 0, i.e. the first message they receive in a block is a 0, they know that they've received the bit b. To figure out what b was, they look back at the last block where the first message they heard was 1, and let $b = 0$ if it was followed by any 0's and $b = 1$ otherwise. Take note of a subtlety here – the listener could have received more than one 1 before the bit b, but this doesn't actually matter as they receive any 0's in the block if and only if $b = 0$.

- The passer (ex-speaker) stays in passer mode, sending only 0's, until they hear a 1 from the other party. This received 1 either confirms the receipt of the role switch signal (a 0 heard before any 1's in a block), or the party already received the switch signal earlier and is already in speaker mode; either way, the passer is free to switch to listener mode in the next block since the other party will be in speaker mode.

In this way, the roles of the speaker and listener have now switched and the protocol is in an entirely "reset" state with one more bit of the noiseless transcript π_0 having been conveyed.

To understand why this protocol is resilient to $\frac{1}{2} - \epsilon$ erasures, we use the following fact: There are at least $\approx n$ blocks with $< \frac{1}{2} - \epsilon$ erasures, and in each such block, there are at least a constant, say 2, pairs of consecutive messages that are unerased. In each such pair of messages, the protocol makes *progress* via one party hearing the other and successfully confirming receipt of the message.

(Formally, these two pairs of unerased consecutive messages are enough to ensure that at least one party completes the goal for that mode and advances modes at the end of the block.)

For a formal description and analysis of this protocol, we refer the reader to Section 5.

3 Preliminaries

All definitions presented in this section are for the binary alphabet $\{0,1\}$.

Notation. In this work, we use the following notations.

- The function $\Delta(x,y)$ represents the Hamming distance between x and y.
- $x[i]$ denotes the i'th bit of a string $x \in \{0,1\}^*$.
- $x||y$ denotes the string x concatenated with the string y.
- The symbol $\perp$ in a message represents the erasure symbol that a party might receive in the erasure model.

3.1 Error Correcting Codes

Definition 3.1 (Error Correcting Code)**.** A family of error correcting codes (ECC) is a family of maps $\mathsf{ECC} = \{\mathsf{ECC}_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}_{n\in\mathbb{N}}$. An ECC has *relative distance* $\alpha > 0$ if for all $n \in \mathbb{N}$ and any $x \neq y \in \{0,1\}^n$,

$$\Delta\left(\mathsf{ECC}_n(x), \mathsf{ECC}_n(y)\right) \geq \alpha p.$$

Binary error correcting codes with relative distance $\approx \frac{1}{2}$ are well known to exist with linear blowup in communication complexity, such that they can be encoded and decoded efficiently.

Theorem 3.2 ([GS00])**.** *For all $\epsilon > 0$, there exists an explicit family of error correcting codes $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \{0,1\}^n \to \{0,1\}^p\}_{n\in\mathbb{N}}$ with relative distance $\frac{1}{2} - \epsilon$ and maximum distance between any two elements $\frac{1}{2}$[4], and with $p = p(n) = O\left(\frac{n}{\epsilon^3}\right)$. Moreover, there are explicit encoding and decoding algorithms that take time $C(\epsilon)n^{O(1)}$ for some constant $C(\epsilon)$. Correctness of decoding holds for up to $\frac{1}{4} - \epsilon$ errors.*

3.2 Noise Resilient Interactive Communication

We formally define a non-adaptive interactive protocol along with error resilience to corruption. The two types of corruptions we will be interested in are erasures and bit flips.

Definition 3.3 (Non-Adaptive Interactive Coding Scheme)**.** A two-party non-adaptive interactive coding scheme π for a function $f(x,y) : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^o$ is an interactive protocol consisting of a fixed number of rounds, denoted $|\pi|$. In each round, a single party fixed beforehand sends a single bit to the other party. At the end of the protocol, each party outputs a guess $\in \{0,1\}^o$.

[4]While the relevant theorem in the paper by Guruswami and Sudan does not directly state that the maximum distance between any two elements is $\frac{1}{2}$, this follows from the fact that their construction is a concatenated code with the inner code as a Hadamard code (for which any two codewords have distance exactly $\frac{1}{2}$).

We say that π is resilient to α fraction of adversarial bit flips (resp. erasures) with probability p if the following holds. For all $x, y \in \{0,1\}^n$, and for all online adversarial attacks consisting of flipping (resp. erasing) at most $\alpha \cdot |\pi|$ of the total rounds, with probability $\geq p$ Alice and Bob both output $f(x, y)$ at the end of the protocol.

4 Binary Interactive Protocol Resilient to $\frac{1}{6}$ Bit Flips

In this section, we present a non-adaptive interactive protocol where Alice and Bob exchange inputs in a way that is resilient to $\frac{1}{6} - \epsilon$ bit flips for any $\epsilon > 0$. This result is optimal; no protocol is resilient to $\geq \frac{1}{6}$ bit flips for all possible functions Alice and Bob might want to compute.

Theorem 4.1 ([EGH16])**.** *For large enough n, there exists a function $f(x, y)$ of Alice and Bob's inputs $x, y \in \{0,1\}^n$, such that any non-adaptive interactive protocol over the binary bit flip channel that computes $f(x, y)$ succeeds with probability at most $\frac{1}{2}$ if a $\frac{1}{6}$ fraction of the transmissions are corrupted.*

As explained in Section 2, in our protocol Alice and Bob each keep track of a string $\hat{y}$ and $\hat{x}$ respectively containing a guess for the other party's input, and a counter c_A and c_B respectively containing their confidence for the current guess. When a party receives a message, they decode it to obtain two parts: the other party's guess and instructions for how to update their own guess. They perform the corresponding updates to their own guess and then send their new guess along with instructions to the other party for how to update their guess. In order to formally state our protocol, we list a few definitions.

4.1 Preliminaries and Definitions

Throughout the protocol, Alice and Bob will send each other instructions, usually denoted δ. To this end, we define two functions for how to use and create the instruction δ.

Definition 4.2 ($(z, c) \oplus \delta$)**.** We define $(z, c) \oplus \delta$ for $z \in \{0,1\}^{\leq n}$, $c \in \mathbb{Z}_{\geq 0}$ and $\delta \in \{0, 1, \leftarrow, \bullet\}$ as the update to (z, c) induced by δ. More specifically, we modify z by the operation δ if $\delta \in \{0, 1, \leftarrow\}$ and increment the counter c if $\delta = \bullet$. That is,

- If $c = 0$ and $\delta \in \{0,1\}$, $|z| < n$, then $(z, c) \oplus \delta := (z||\delta, c)$.
- If $c = 0$ and $\delta = \leftarrow$, $|z| > 0$, then $(z, c) \oplus \leftarrow := (z[1 : |z| - 1], c)$.
- If $c > 0$ and $\delta \in \{0, 1, \leftarrow\}$, then $(z, c) \oplus \leftarrow := (z, c - 1)$.
- If $\delta = \bullet$, then $(z, c) \oplus \bullet := (z, c + 1)$.
- Otherwise, $(z, c) \oplus \delta := (z, c)$.

Definition 4.3 (op_z)**.** We define $\mathsf{op}_z(z')$ for $z \in \{0,1\}^{\leq n}$ to be the instruction that brings z' one bit closer to z (or $\bullet$ if $z' = z$). That is,

- If z' is a strict prefix of z, then $\mathsf{op}_z(z') := z[|z'| + 1]$.
- If z' is not a prefix of z, then $\mathsf{op}_z(z') := \leftarrow$.

- If $z' = z$, then $\mathsf{op}_z(z') := \bullet$.

Note that $(z', c) \oplus \mathsf{op}_z(z')$ either increases c by 1 if $z = z'$, and otherwise either decreases c by 1 if $c > 0$ or changes z' to be one character closer to z.

Next, we define the error correcting code family ECC that Alice and Bob use in the protocol, and show that the desired ECC with the listed properties exists. For shorthand, we will denote the domain of the ECC as

$$\Sigma = \{0,1\}^{\le n} \times \{0,1\}^{\le n} \times \{0, 1, \leftarrow, \bullet\}$$

throughout the section.

Definition 4.4 (ECC)**.** For a given $\epsilon > 0$, we define the error correcting code family

$$\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \Sigma \to \{0,1\}^{M(n)}\}_{n\in\mathbb{N}}$$

with the following properties:

- $M = O_\epsilon(n)$.
- For any $n \in \mathbb{N}$ and for any $(z_0, z_0') \neq (z_1, z_1') \in \{0,1\}^{\le n} \times \{0,1\}^{\le n}$ and $\delta_0, \delta_1 \in \{0, 1, \leftarrow, \bullet\}$,
$$\Delta\big(\mathsf{ECC}_{\epsilon,n}(z_0, z_0', \delta_0), \mathsf{ECC}(z_1, z_1', \delta_1)\big) \ge \left(\frac{1}{2} - \epsilon\right) \cdot M, \tag{1}$$
- For any $n \in \mathbb{N}$ and for any $z, z' \in \{0,1\}^{\le n} \times \{0,1\}^{\le n}$ and $\delta_0 \neq \delta_1 \in \{0, 1, \leftarrow, \bullet\}$,
$$\Delta\big(\mathsf{ECC}_{\epsilon,n}(z, z', \delta_0), \mathsf{ECC}(z, z', \delta_1)\big) \ge \frac{2}{3}M. \tag{2}$$
- Decoding up to $\frac{1}{6} - \epsilon$ errors can be done in time $C(\epsilon)n^{O(1)}$ for some constant $C(\epsilon)$.

Claim 4.5. *For all $\epsilon > 0$, an explicit error correcting code family ECC_ϵ from Definition 4.4 exists. In other words, there exists an explicit error correcting code family that simultaneously satisfies all the properties listed.*

Proof. For a fixed $n \in \mathbb{N}$, let $\mathsf{ECC}' : \{0,1\}^{\le n} \times \{0,1\}^{\le n} \to \{0,1\}^{M/3}$ be an efficiently encodable and decodable error correcting code with relative distance in the range $\left[\frac{1}{2} - \epsilon, \frac{1}{2}\right]$ between any pair of codewords, where correctness of decoding holds for $\le \frac{1}{4} - \epsilon$ errors. This exists for some $M = O_\epsilon(n)$ by Theorem 3.2. Let DEC' be the corresponding decoding algorithm. We define

$$\mathsf{ECC}_{\epsilon,n}(z, z', \delta') = \begin{cases} \overline{\mathsf{ECC}'(z,z')}||\mathsf{ECC}'(z,z')||\mathsf{ECC}'(z,z'), & \delta' = 0 \\ \mathsf{ECC}'(z,z')||\overline{\mathsf{ECC}'(z,z')}||\mathsf{ECC}'(z,z'), & \delta' = 1 \\ \mathsf{ECC}'(z,z')||\mathsf{ECC}'(z,z')||\overline{\mathsf{ECC}'(z,z')}, & \delta' = \leftarrow \\ \overline{\mathsf{ECC}'(z,z')}||\overline{\mathsf{ECC}'(z,z')}||\overline{\mathsf{ECC}'(z,z')}, & \delta' = \bullet, \end{cases}$$

where $\bar{s}$ denotes the bitwise not of string s. Then Equation (1) holds because for any $(z_0, z_0') \neq (z_1, z_1')$, the relative distance between $\mathsf{ECC}'(z_0, z_0')$ or $\overline{\mathsf{ECC}'(z_0, z_0')}$ to $\mathsf{ECC}'(z_1, z_1')$ or $\overline{\mathsf{ECC}'(z_1, z_1')}$ is $\ge \frac{1}{2} - \epsilon$, and Equation (2) holds because for fixed z, z' the four codewords for $\delta' = 0, 1, \leftarrow, \bullet$ are distance $\frac{2}{3}$ apart.

It remains to show if a codeword $\mathsf{ECC}_\epsilon(z, z', \delta')$ is corrupted on $\le \left(\frac{1}{6} - \epsilon\right)$ locations to a string $s \in \{0,1\}^M$, then it can be correctly and efficiently decoded. Our decoding algorithm $\mathsf{DEC}(s)$ is as follows:

For a string $s = s_1 s_2 s_3$ where s_i are each length $\frac{M}{3}$, consider each of the following strings:

$$\begin{aligned} S_0 &= \overline{s_1}||s_2||s_3 \\ S_1 &= s_1||\overline{s_2}||s_3 \\ S_\leftarrow &= s_1||s_2||\overline{s_3} \\ S_\bullet &= \overline{s_1}||\overline{s_2}||\overline{s_3}. \end{aligned}$$

For each $\delta \in \{0, 1, \leftarrow, \bullet\}$, denote $S_\delta = s_1^\delta||s_2^\delta||s_3^\delta$ and consider the string

$$S'_\delta = \text{maj}(s_1^\delta[1], s_2^\delta[1], s_3^\delta[1])||\dots||\text{maj}(s_1^\delta[M/3], s_2^\delta[M/3], s_3^\delta[M/3]) \in \{0,1\}^{M/3}.$$

Compute $(z_\delta, z'_\delta) \leftarrow \mathsf{DEC}'(S'_\delta)$, and let $d_\delta = \frac{1}{M}\Delta(\mathsf{ECC}(z_\delta, z'_\delta, \delta), s)$. If $d_\delta \le \left(\frac{1}{6} - \epsilon\right)$, output $(z_\delta, z'_\delta, \delta)$.

If $d_\delta > \left(\frac{1}{6} - \epsilon\right)$ for all $\delta \in \{0, 1, \leftarrow, \bullet\}$, then output $\emptyset$.

We now show correctness of this decoding algorithm if there are fewer than $\frac{1}{6} - \epsilon$ corruptions. First note that for any string s, since any two codewords $\mathsf{ECC}(z_{\delta_0}, z'_{\delta_0}, \delta_0), \mathsf{ECC}(z_{\delta_1}, z'_{\delta_1}, \delta_1)$ with $\delta_0 \neq \delta_1$ are at least relative distance $\frac{1}{2} - \epsilon$ apart, d_δ can be $\le \frac{1}{6} - \epsilon$ for at most one value of δ. This means that the output condition is satisfied for at most one codeword of the four.

If s is relative distance $\le \frac{1}{6} - \epsilon$ to a codeword $\mathsf{ECC}_\epsilon(z, z', \delta')$, then $S_{\delta'}$ is relative distance $\le \frac{1}{6} - \epsilon$ to $\mathsf{ECC}'(z, z')||\mathsf{ECC}'(z, z')||\mathsf{ECC}'(z, z')$. This means that $S'_{\delta'}$ and $\mathsf{ECC}(z, z')$ differ on at most $\frac{1}{2} \cdot (\frac{1}{6} - \epsilon) \cdot M$ locations, which is $< \frac{1}{4} - \epsilon$ fraction of the $\frac{M}{3}$ locations, since the adversary must corrupt two out of three values $s_1^{\delta'}[j], s_2^{\delta'}[j], s_3^{\delta'}[j]$ in order for $S'_{\delta'}[j] \neq \mathsf{ECC}(z, z')[j]$. Thus, $(z, z') = \mathsf{DEC}'(S'_{\delta'})$, and the decoding algorithm outputs (z, z', δ').

□

For the rest of the section, ECC will denote the error correcting code $\mathsf{ECC}_{\epsilon,n}$ as defined in Definition 4.4 when n and ϵ are clear.

4.2 Formal Protocol

We are now ready to formally state the protocol.

Protocol 1 : Randomized Protocol Resilient to $\frac{1}{6} - 2\epsilon$ Corruptions

Fix $n \in \mathbb{N}, \epsilon > 0$. Suppose Alice and Bob's private inputs are $x, y \in \{0,1\}^n$ respectively, such that $y[1] = 0$. The protocol consists of $T = \frac{n}{\epsilon}$ (assume T is even) messages numbered $1, \dots, T$, each of length M. Alice sends the odd messages and Bob sends the even. For our convenience, Alice and Bob both agree that the 0'th message is $\mathsf{ECC}(\emptyset, \emptyset, 0)$, sent uncorrupted by Bob to Alice.

Alice and Bob track a private guess for the other party's input, denoted $\hat{y} \in \{0,1\}^{\le n}$ and $\hat{x} \in \{0,1\}^{\le n}$ respectively, both initialized at $\emptyset$ at the beginning of the protocol. They also each have a personal counter c_A and c_B respectively, containing their confidence in their current guess $\hat{y}$ or $\hat{x}$ respectively, initialized at 0.

In what follows, we describe Alice's behavior. Bob's behavior is identical, except replacing $\hat{y}, c_A$ by $\hat{x}, c_B$ and notationally switching x and y in general (notably sending $\mathsf{ECC}(\hat{x}, y^*, \delta^*)$ instead of $\mathsf{ECC}(\hat{y}, x^*, \delta^*)$). At the end of the protocol, Alice and Bob will output $\hat{y}$ and $\hat{x}$ respectively.

Alice

Alice has just received a message m from Bob. She first tries to set $(x^*, y^*, \delta^*) \in \Sigma$ and $p^* \in [0,1]$ as follows. If no condition is satisfied, we say that she does not set (x^*, y^*, δ^*). In what follows, we let $d_m(x', y', \delta')$ denote $\frac{1}{M} \cdot \Delta(m, \mathsf{ECC}(x', y', \delta'))$.

1. She sets $(x^*, y^*, \delta^*) \leftarrow (x, \hat{y}, \delta)$ if $d_m(x, \hat{y}, \delta) < \frac{1}{3}$ by checking all $\delta \in \{0, 1, \leftarrow, \bullet\}$ individually. If she sets (x^*, y^*, δ^*) in this way, she also sets
$$p^* = 1 - 3d_m(x, \hat{y}, \delta).$$
Note that $d_m(x, \hat{y}, \delta) < \frac{1}{3}$ for at most one δ, since the codewords $\mathsf{ECC}(x, \hat{y}, \delta)$ all have relative distance $\frac{2}{3}$.
2. Otherwise, if $\mathsf{DEC}(m) \neq \emptyset$, she sets $(x^*, y^*, \delta^*) \leftarrow \mathsf{DEC}(m)$. She sets
$$p^* = 0.5 - 3d_m(x^*, y^*, \delta^*)$$
Recall that $\mathsf{DEC}(m) \neq \emptyset$ only if $d_m(x^*, y^*, \delta^*) \leq \frac{1}{6} - \epsilon$.

Next, based on whether or not Alice set (x^*, y^*, δ^*), she does the following:

- If she set (x^*, y^*, δ^*), she does the following update with probability p^* (doing nothing if no condition is satisfied):
  - If $y^* = \hat{y}$, then $(\hat{y}, c_A) \leftarrow (\hat{y}, c_A) \oplus \delta^*$.
  - If $y^* \neq \hat{y}$ and $\delta^* = \bullet$, then $(\hat{y}, c_A) \leftarrow (\hat{y}, c_A) \oplus \mathsf{op}_{y^*}(\hat{y})$.

  She sends $\mathsf{ECC}(\hat{y}, x^*, \mathsf{op}_x(x^*))$.
- Otherwise, if she did not sent (x^*, y^*, δ^*), she does nothing to $\hat{y}, c_A$ and sends $\mathsf{ECC}(\hat{y}, x, \bullet)$.

4.3 Analysis

We now state our main error resilience theorem.

Theorem 4.6. *Protocol 1 is resilient to a* $\left(\frac{1}{6} - 2\epsilon\right)$*-fraction of errors with probability* $1 - 2\exp\left(\frac{-\epsilon n}{100}\right)$.[5]

To prove Theorem 4.6, we analyze the effects of corruption on the *good* and *bad updates* Alice/Bob make. We begin by defining good and bad updates. After receiving a message from Bob, Alice updates (with some probability) her values of c_A and $\hat{y}$ (if these values both stay the same, we say that Alice did not make an update). We say that an update is *good* if her new value of $(\hat{y}, c_A)$ is $(\hat{y}, c_A) \oplus \mathsf{op}_y(\hat{y})$. An update is otherwise *bad.* We similarly define good and bad updates for Bob. For each $t \in [0, \ldots, T]$, we define the following potential functions:

- ψ_t^A is defined to be the total number of good updates minus the number of bad updates Alice has done in response to messages $0, \ldots, t$.
- ψ_t^B is defined to be the total number of good updates minus the number of bad updates Bob has done in response to messages $0, \ldots, t$.

[5]We only consider ϵ sufficiently small (say < 0.1).

For instance, since there's an agreed-upon 0'th message from Bob to Alice, $\mathbb{E}[\psi_0^A] = 0.5$ and $\psi_0^B = 0$. For $t = -1$, we let $\psi_{-1}^A = \psi_{-1}^B = 0$.

Lemma 4.7. *After message t, $\psi_t^A \geq n$ if and only if Alice's value of $\hat{y}$ is equal to y, and $\psi_t^B \geq n$ if and only if Bob's value of $\hat{x}$ is equal to x.*

In particular, at the end of the protocol, Alice outputs $\hat{y} = y$ if and only if $\psi_T^A \geq n$, and Bob outputs $\hat{x}$ if and only if $\psi_T^B \geq n$.

Proof. We prove this for Alice, as the proof for Bob is identical. We keep track of the number of good updates Alice must do to have $\hat{y} = y$. At the beginning of the protocol, since $\hat{y} = \emptyset$, Alice needs to perform n good updates (appending the n bits of y) so that $\hat{y} = y$. We show that any good update decreases this number by 1, and any bad update increases this number by 1. Then, at any point, $\hat{y} = y$ if and only if the number of good updates minus the number of bad updates is at least n.

To show that every bad update increases this number by 1, we show that a bad update (i.e., any update other than $(\hat{y}, c_A) \leftarrow (\hat{y}, c_A) \oplus \mathsf{op}_y(\hat{y})$), when followed by the good update $(\hat{y}, c_A) \leftarrow (\hat{y}, c_A) \oplus \mathsf{op}_y(\hat{y})$, results back in the original value of $(\hat{y}, c_A)$. If the bad update appends a bit to $\hat{y}$, then the new value of $\hat{y}$ must not be a prefix of y. Then the good update $\leftarrow$ undoes this. If the bad update deletes the last bit of $\hat{y}$ incorrectly, then re-appending this bit undoes this. If the bad update increases c_A incorrectly, then $\hat{y} \neq y$, so the next good update is $\mathsf{op}_y(\hat{y}) \neq \bullet$ which causes c_A to decrease by 1. If the bad update decreases c_A incorrectly, then $\hat{y} = y$, and the next good update is $\mathsf{op}_y(\hat{y}) = \bullet$ which increases c_A by 1.

□

For Alice, we group each consecutive pair of Alice-to-Bob and Bob-to-Alice messages (i.e., messages $2k-1$ and $2k$). For Bob, we group each consecutive pair of Bob-to-Alice and Alice-to-Bob messages (i.e., messages $2k-2$ and $2k-1$), starting with the unsent message 0 which we recall is understood to be $\mathsf{ECC}(\emptyset, \emptyset, 0)$. We define the following potential function for each $k \in [0, T/2]$:

- $\Psi_k^A = \psi_{2k}^A + \min\{\psi_{2k}^B + \min\{\rho_{2k+1}^B, 0.5\}, n\}$
- $\Psi_k^B = \psi_{2k-1}^B + \min\{\psi_{2k-1}^A + \min\{\rho_{2k}^A, 0.5\}, n\}$,

where ρ_t^A and ρ_t^B are defined as follows:

- For even t, ρ_t^A is the expected number of good updates minus the number of bad updates that Alice will do in response to message t, given messages $1, \dots, t-1$, if message t from Bob is uncorrupted.
- For odd t, ρ_t^B is the expected number of good updates minus the number of bad updates that Bob will do in response to message t, given messages $1, \dots, t-1$, if message t from Alice is uncorrupted. For instance, since Alice sends $\mathsf{ECC}(\hat{y}, \emptyset, x[1])$ as her first message, $\mathbb{E}[\rho_1^B] = 0.5$.

Lemma 4.8. *If an α_{2k-1} fraction of message $2k-1$ and an α_{2k} fraction of message $2k$ is corrupted, then*

$$\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}. \tag{3}$$

If an α_{2k-2} fraction of message $2k-2$ and an α_{2k-1} fraction of message $2k-1$ is corrupted, then

$$\mathbb{E}[\Psi_k^B - \Psi_{k-1}^B] \geq 1 - 6\epsilon - 3\alpha_{2k-2} - 3\alpha_{2k-1}. \tag{4}$$

We delay the proof of Lemma 4.8 to Section 4.3.1 as it is quite long and instead use it to complete the proof of Theorem 4.6.

Proof of Theorem 4.6. Consider any adversarial corruption of Protocol 1 consisting of fewer than $\frac{1}{6} - 2\epsilon$ corruptions. We will show that Alice and Bob must both output the other party's input correctly. Let $\alpha_1, \dots, \alpha_T$ denote the fractional number of corruptions in messages $1, \dots, T$, such that $\alpha_1 + \dots + \alpha_T < \left(\frac{1}{6} - 2\epsilon\right) \cdot T$. By default, we let $\alpha_0 = 0$. For $k \in [T/2]$, we define the random variables

$$\Phi_k^A = \Psi_k^A - k + 6k\epsilon + \sum_{i=0}^{2k} 3\alpha_i,$$
$$\Phi_k^B = \Psi_k^B - k + 6k\epsilon + \sum_{i=0}^{2k-1} 3\alpha_i.$$

Then, $\Phi_0^A = \Psi_0^A \in \{0.5, 1.5\}$ and $\Phi_0^B = \Psi_0^B = 0.5$.

By Lemma 4.8,

$$\begin{aligned} \mathbb{E}[\Phi_k^A] &= \mathbb{E}\left[\Psi_k^A - k + 6k\epsilon + \sum_{i=0}^{2k} 3\alpha_i\right] \\ &\geq \mathbb{E}\left[\Psi_{k-1}^A - (k-1) + 6(k-1)\epsilon + \sum_{i=0}^{2k-2} 3\alpha_i\right] \\ &= \mathbb{E}[\Phi_{k-1}^A], \end{aligned}$$

so Φ_k^A is a submartingale with bounded distance

$$\begin{aligned} |\Phi_k^A - \Phi_{k-1}^A| &= \left|\Psi_k^A - \Psi_{k-1}^A - 1 + 6\epsilon + 3\alpha_{2k-1} + 3\alpha_{2k}\right| \\ &\leq \left|U_{2k}^A\right| + \left|U_{2k-1}^B\right| + \left|\rho_{2k+1}^B\right| + \left|\rho_{2k-1}^B\right| + \left|-1 + 6\epsilon + 3\alpha_{2k-1} + 3\alpha_{2k}\right| \\ &< 10, \end{aligned}$$

where U_{2k}^A is $+1$, -1, or 0 if Alice made a good, bad, or no update respectively in response to message $2k$, and U_{2k}^B is defined the same way but for Bob in response to message $2k-1$. Similarly, Φ_k^B is a submartingale with bounded distance < 10.

Note that if $\Psi_{T/2}^A \geq 2n$, then $\psi_T^A = \Psi_{T/2}^A - \min\{\psi_T^B, n\} \geq n$, meaning by Lemma 4.7 that Alice holds $\hat{y} = y$ at the end of the protocol. Thus, by Azuma's inequality, the probability that Alice outputs correctly at the end of the protocol is at least

$$\begin{aligned} \Pr\left[\Psi_{T/2}^A \geq 2n\right] &= 1 - \Pr\left[\Phi_{T/2}^A - \Phi_0^A < 2n - \frac{T}{2} + 3T\epsilon + \sum_{i=0}^{T} 3\alpha_i - \Phi_0^A\right] \\ &\geq 1 - \Pr\left[\Phi_{T/2}^A - \Phi_0^A < 2n - \frac{T}{2} + 3T\epsilon + 3 \cdot \left(\frac{1}{6} - 2\epsilon\right) \cdot T\right] \\ &\geq 1 - \Pr\left[\Phi_{T/2}^A - \Phi_0^A < -n\right] \\ &\geq 1 - \exp\left(\frac{-\epsilon n}{100}\right). \end{aligned}$$

The same calculation for Bob shows that Bob outputs correctly at the end of the protocol with probability at least $1-\exp\left(\frac{-\epsilon n}{100}\right)$ as well. Then by a union bound, the probability that both parties output correctly is at least

$$1-2\cdot\exp\left(\frac{-\epsilon n}{100}\right).$$

□

4.3.1 Proof of Lemma 4.8

We only prove Inequality (3), as the proof for Inequality (4) is identical.

Define Ψ_k to be $\psi^A_{2k}+\rho^A_{2k}+\min\{\psi^B_{2k-1},n\}$. We will first determine the value of $\mathbb{E}[\Psi^A_k-\Psi_k]$, which we will then use to complete the proof of the lemma.

Claim 4.9. *If $\psi^B_{2k-1}<n$, then*

$$\mathbb{E}[\Psi^A_k-\Psi_k]\geq 0.5-3\epsilon-3\alpha_{2k}. \tag{5}$$

If $\psi^B_{2k-1}\geq n$, then

$$\mathbb{E}[\Psi^A-\Psi_k]\geq -3\epsilon-3\alpha_{2k}. \tag{6}$$

Proof. Define the random variable U^A_{2k} to be 1 if Alice makes a good update, -1 if she makes a bad update, and 0 if she makes no update in response to message $2k$. Let $\mathsf{ECC}(x_{2k},y_{2k},\delta_{2k})$ be Bob's intended message for message $2k$, and let m_{2k} be the message Alice receives. Let $(x^*_{2k},y^*_{2k},\delta^*_{2k})\in\Sigma$ with a corresponding p^*_{2k} be what Alice computes in her protocol, if they exist. Moreover, let $d^*_{2k}=d_{m_{2k}}(x^*_{2k},y^*_{2k},\delta^*_{2k})$. In order to show Equations (5) and (6), we will show they hold for any value of $(x_{2k},y_{2k},\delta_{2k})$, which implies they hold in expectation.

Proof of Equation (5). We first show that if $\psi^B_{2k-1}<n$, then $\mathbb{E}[\Psi^A_k-\Psi_k]\geq 0.5-3\epsilon-3\alpha_{2k}$. To start, we have

$$\begin{aligned}\mathbb{E}[\Psi^A_k-\Psi_k]&=\mathbb{E}\left[\psi^A_{2k}+\min\{\psi^B_{2k}+\min\{\rho^B_{2k+1},0.5\},n\}-\psi^A_{2k-1}-\rho^A_{2k}-\min\{\psi^B_{2k-1},n\}\right]\\&=\mathbb{E}\left[\psi^A_{2k}+\psi^B_{2k}+\min\{\rho^B_{2k+1},0.5\}-\psi^A_{2k-1}-\rho^A_{2k}-\psi^B_{2k-1}\right]\\&=\mathbb{E}[U^A_{2k}-\rho^A_{2k}+\min\{\rho^B_{2k+1},0.5\}],\end{aligned}$$

as $\psi^B_{2k-1}=\psi^B_{2k}<n$, implying that $\psi^B_{2k}+\min\{\rho^B_{2k+1},0.5\},\psi^B_{2k-1}\leq n$.

We split the analysis into cases. In each case, we bound the three values $\mathbb{E}[U^A_{2k}],\rho^A_{2k},\rho^B_{2k+1}$ separately, and combine them to bound $\mathbb{E}[\Psi^A_k-\Psi_k]$. Since $\psi^B_{2k-1}<n$, by Lemma 4.7, Bob's value of $\hat{x}$ after receiving message $2k-1$ is not x, i.e. $x_{2k}\neq x$. Thus, $\rho^A_{2k}\leq 0.5$. Furthermore, $\rho^B_{2k+1}\geq 0$ because any uncorrupted message Alice sends cannot cause Bob to perform a bad update. If no other constraint on ρ^A_{2k} or $\min\{\rho^B_{2k+1},0.5\}$ is used in the final calculation, we omit its computation.

Case 1: $(x^*_{2k},y^*_{2k},\delta^*_{2k})$ does not exist.

- $\mathbb{E}[U^A_{2k}]=0$ because Alice does not update.
- $\rho^B_{2k+1}\geq 0.5$ because she replies with $\mathsf{ECC}(\hat{y},x,\bullet)$, and $(\hat{x},c_B)\oplus\mathsf{op}_x(\hat{x})$ is a positive update.

- $\alpha_{2k} \geq \left(\frac{1}{6} - \epsilon\right)$ or we would have $(x_{2k}, y_{2k}, \delta_{2k}) = (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$.

$$\begin{aligned} \implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k} + \min\{\rho^B_{2k+1}, 0.5\}] \\ &\geq 0 - 0.5 + 0.5 = 0 \\ &\geq 0.5 - 3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 2: $(x_{2k}, y_{2k}, \delta_{2k}) = (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* $(y_{2k} = \hat{y}$ *or* $\delta_{2k} = \bullet)$*.*

- $\mathbb{E}[U^A_{2k}] \geq 0.5 - 3\alpha_{2k}$. Alice makes a good update with probability $p^*_{2k} = 0.5 - 3d^*_{2k} > 0$, and $d^*_{2k} \leq \alpha_{2k}$, so $\mathbb{E}[U^A_{2k}] = 0.5 - 3d^*_{2k} \geq 0.5 - 3\alpha_{2k}$.
- $\rho^B_{2k+1} \geq 0.5$ because she replies with $\mathsf{ECC}(\hat{y}, x_{2k}, \mathsf{op}_x(x_{2k}))$, and $(\hat{x} = x_{2k}, c_B) \oplus \mathsf{op}_x(x_{2k})$ is a positive update since $x_{2k} = \hat{x}$.

$$\begin{aligned} \implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k} + \min\{\rho^B_{2k+1}, 0.5\}] \\ &\geq (0.5 - 3\alpha_{2k}) - 0.5 + 0.5 \\ &\geq 0.5 - 3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 3: $(x_{2k}, y_{2k}, \delta_{2k}) = (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* $(y_{2k} \neq \hat{y}$ *and* $\delta_{2k} \neq \bullet)$*.*

- $U^A_{2k} = 0$ because Alice will never change $(\hat{y}, c_A)$ in response to $(x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ with the given constraints.
- $\rho^A_{2k} = 0$ because again Alice will never change $(\hat{y}, c_A)$ in response to $(x_{2k}, y_{2k}, \delta_{2k})$.
- $\rho^B_{2k+1} \geq 0.5$ because she replies with $\mathsf{ECC}(\hat{y}, x_{2k}, \mathsf{op}_x(x_{2k}))$.

$$\begin{aligned} \implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k} + \min\{\rho^B_{2k+1}, 0.5\}] \\ &\geq 0 - 0 + 0.5 \\ &\geq 0.5 - 3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 4: $(x_{2k}, y_{2k}, \delta_{2k}) \neq (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* $(x^*_{2k} = x$ *and* $y^*_{2k} = \hat{y})$*.*

- $\mathbb{E}[U^A_{2k}] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$. Alice makes a bad update with probability at most $p^*_{2k} = 1 - 3d^*_{2k} > 0$ ("at most" because she may make a good update or no update). Substituting $d^*_{2k} \geq (\frac{1}{2} - \epsilon) - \alpha_{2k}$ which follows from Equation (1) gives the desired result.
- $\rho^B_{2k+1} \geq 0.5$ because she replies with $\mathsf{ECC}(\hat{y}, x, \bullet)$.

$$\begin{aligned} \implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k} + \min\{\rho^B_{2k+1}, 0.5\}] \\ &\geq (0.5 - 3\epsilon - 3\alpha_{2k}) - 0.5 + 0.5 \\ &= 0.5 - 3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 5: $(x_{2k}, y_{2k}, \delta_{2k}) \neq (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* $(x^*_{2k} \neq x$ *or* $y^*_{2k} \neq \hat{y})$*.*

- $\mathbb{E}[U_{2k}^A] \geq 1 - 3\epsilon - 3\alpha_{2k}$. Alice makes a bad update with probability at most $p_{2k}^* = 0.5 - 3d_{2k}^*$. Substituting $d_{2k}^* \geq \left(\frac{1}{2} - \epsilon\right) - \alpha_{2k}$ gives the desired result.

$$\begin{aligned} \implies \mathbb{E}[\Psi_k^A - \Psi_k] &= \mathbb{E}[U_{2k}^A - \rho_{2k}^A + \min\{\rho_{2k+1}^B, 0.5\}] \\ &\geq (1 - 3\epsilon - 3\alpha_{2k}) - 0.5 + 0 \\ &= 0.5 - 3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Proof of Equation (6). Next, we show that if $\psi_{2k-1}^B \geq n$, then $\mathbb{E}[\Psi_k^A - \Psi_k] \geq -3\alpha_{2k}$. Since $\psi_{2k-1}^B = \psi_{2k}^B \geq n$, we have that

$$\begin{aligned} \mathbb{E}[\Psi_k^A - \Psi_k] &= \mathbb{E}\left[\psi_{2k}^A + \min\{\psi_{2k}^B + \min\{\rho_{2k+1}^B, 0.5\}, n\} - \psi_{2k-1}^A - \rho_{2k}^A - \min\{\psi_{2k-1}^B, n\}\right] \\ &= \mathbb{E}[U_{2k}^A - \rho_{2k}^A]. \end{aligned}$$

Again, we split the analysis into cases. In each case we compute $\mathbb{E}[U_{2k}^A]$ and ρ_{2k}^A separately. Since $\psi_{2k-1}^B \geq n$, by Lemma 4.7, $x_{2k} = x$. We have that $\rho_{2k}^A \leq 1$ (Alice cannot perform more than one good update in response to any possible message sent by Bob).

Case 1: $(x_{2k}^*, y_{2k}^*, \delta_{2k}^*)$ does not exist.

- $\mathbb{E}[U_{2k}^A] = 0$ because Alice does not update.
- $\rho_{2k}^A \leq 3\epsilon + 3\alpha_{2k}$. If $y_{2k} = \hat{y}$, then $\rho_{2k}^A \leq 1$ and $\alpha_{2k} > \frac{1}{3}$ since otherwise $(x_{2k}^*, y_{2k}^*, \delta_{2k}^*)$ would've been $(x_{2k}, y_{2k}, \delta_{2k})$. Else if $y_{2k} \neq \hat{y}$, $\rho_{2k}^A \leq 0.5$ and $\alpha_{2k} > \left(\frac{1}{6} - \epsilon\right)$. Regardless, the result follows.

$$\begin{aligned} \implies \mathbb{E}[\Psi_k^A - \Psi_k] &= \mathbb{E}[U_{2k}^A - \rho_{2k}^A] \\ &\geq 0 - 3\epsilon - 3\alpha_{2k} \\ &\geq -3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 2: $(x_{2k}, y_{2k}, \delta_{2k}) = (x_{2k}^*, y_{2k}^*, \delta_{2k}^*)$ and $y_{2k} = \hat{y}$.

- $\mathbb{E}[U_{2k}^A] \geq 1 - 3\alpha_{2k}$, because Alice makes a good update with probability $p_{2k}^* = 1 - 3d_{2k}^*$ and $\alpha_{2k} \geq d_{2k}^*$.

$$\begin{aligned} \implies \mathbb{E}[\Psi_k^A - \Psi_k] &= \mathbb{E}[U_{2k}^A - \rho_{2k}^A] \\ &\geq 1 - 3\alpha_{2k} - 1 \\ &\geq -3\epsilon - 3\alpha_{2k}. \end{aligned}$$

Case 3: $(x_{2k}, y_{2k}, \delta_{2k}) = (x_{2k}^*, y_{2k}^*, \delta_{2k}^*)$ and ($y_{2k} \neq \hat{y}$ and $\delta_{2k} = \bullet$).

- $\mathbb{E}[U_{2k}^A] \geq 0.5 - 3\alpha_{2k}$, because Alice makes a good update with probability $p_{2k}^* = 0.5 - 3d_{2k}^*$ and $\alpha_{2k} \geq d_{2k}^*$.

- $\rho^A_{2k} \leq 0.5$ since $y_{2k} \neq \hat{y}$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k}] \\ &\geq 0.5 - 3\alpha_{2k} - 0.5 \\ &\geq -3\epsilon - 3\alpha_{2k}.\end{aligned}$$

Case 4: $(x_{2k}, y_{2k}, \delta_{2k}) = (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* ($y_{2k} \neq \hat{y}$ *and* $\delta_{2k} \neq \bullet$)*.*

- $U^A_{2k} = 0$ because Alice will never change $(\hat{y}, c_A)$ in response to $(x^*_{2k}, y^*_{2k}, \delta^*_{2k})$.
- $\rho^A_{2k} = 0$ for the same reason.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k}] \\ &\geq 0 - 0 \\ &\geq -3\epsilon - 3\alpha_{2k}.\end{aligned}$$

Case 5: $(x_{2k}, y_{2k}, \delta_{2k}) \neq (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* ($x^*_{2k} = x$ *and* $y^*_{2k} = \hat{y} = y_{2k}$)*.*

- $\mathbb{E}[U^A_{2k}] \geq 1 - 3\alpha_{2k}$. Alice makes a bad update with probability $p^*_{2k} = 1 - 3d^*_{2k} > 0$. Substituting $d^*_{2k} \geq \frac{2}{3} - \alpha_{2k}$ gives the desired result.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k}] \\ &\geq (1 - 3\alpha_{2k}) - 1 \\ &\geq -3\epsilon - 3\alpha_{2k}.\end{aligned}$$

Case 6: $(x_{2k}, y_{2k}, \delta_{2k}) \neq (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* ($x^*_{2k} = x$ *and* $y^*_{2k} = \hat{y} \neq y_{2k}$)*.*

- $\mathbb{E}[U^A_{2k}] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$. Alice makes a bad update with probability at most $\leq p^*_{2k} = 1 - 3d^*_{2k} > 0$. Substituting $d^*_{2k} \geq (\frac{1}{2} - \epsilon) - \alpha_{2k}$ gives the desired result.
- $\rho^A_{2k} \leq 0.5$ since $y_{2k} \neq \hat{y}$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k}] \\ &\geq (0.5 - 3\epsilon - 3\alpha_{2k}) - 0.5 \\ &= -3\epsilon - 3\alpha_{2k}.\end{aligned}$$

Case 7: $(x_{2k}, y_{2k}, \delta_{2k}) \neq (x^*_{2k}, y^*_{2k}, \delta^*_{2k})$ *and* ($x^*_{2k} \neq x$ *or* $y^*_{2k} \neq \hat{y}$)*.*

- $\mathbb{E}[U^A_{2k}] \geq 1 - 3\epsilon - 3\alpha_{2k}$. Alice makes a bad update with probability at most $p^*_{2k} = \frac{1}{2} - 3d^*_{2k} > 0$. Substituting $d^*_{2k} \geq (\frac{1}{2} - \epsilon) - \alpha_{2k}$ gives the desired result.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi_k] &= \mathbb{E}[U^A_{2k} - \rho^A_{2k}] \\ &\geq (1 - 3\epsilon - 3\alpha_{2k}) - 1 \\ &= -3\epsilon - 3\alpha_{2k}.\end{aligned}$$

□

We return now to the proof of Lemma 4.8. Recall that we want to show

$$\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.$$

We use Claim 4.9 to assist us in calculating $\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A]$. Define U_{2k-1}^B to be 1 if Bob makes a good update, -1 if he makes a bad update, and 0 if he makes no update in response to message $2k-1$ from Alice. Let $\mathsf{ECC}(y_{2k-1}, x_{2k-1}, \delta_{2k-1})$ be Alice's intended message for message $2k$, and let m_{2k-1} be the message Bob receives. Let $(x_{2k-1}^*, y_{2k-1}^*, \delta_{2k-1}^*) \in \Sigma$ with a corresponding p_{2k-1}^* be what Bob computes in his protocol, if they exist. Let $d_{2k-1}^* = d_{m_{2k-1}}(y_{2k-1}^*, x_{2k-1}^*, \delta_{2k-1}^*)$. Note that the triple $(y_{2k-1}, x_{2k-1}, \delta_{2k-1})$ is not necessarily deterministic, but we will show that Equations (3) and (4) hold for any specific value of $(y_{2k-1}, x_{2k-1}, \delta_{2k-1})$.

The strategy is to note that

$$\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] = \mathbb{E}[(\Psi_k - \Psi_{k-1}^A) + (\Psi_k^A - \Psi_k)].$$

We have already calculated $\mathbb{E}[\Psi_k^A - \Psi_k]$ in Claim 4.9, so it remains to analyze the term $\Psi_k - \Psi_{k-1}^A$.

We split the analysis into two cases based on the value of ψ_{2k-2}^B.

Proof of Lemma 4.8 when $\psi_{2k-2}^B < n$. In this case, we have that

$$\begin{aligned}\Psi_k - \Psi_{k-1}^A &= \psi_{2k-1}^A + \rho_{2k}^A + \min\{\psi_{2k-1}^B, n\} - \psi_{2k-2}^A - \min\{\psi_{2k-2}^B + \min\{\rho_{2k-1}^B, 0.5\}, n\} \\ &= \rho_{2k}^A + U_{2k-1}^B - \min\{\rho_{2k-1}^B, 0.5\},\end{aligned}$$

since $\psi_{2k-1}^A = \psi_{2k-2}^A$ and $\rho_{2k-2} + \min\{\rho_{2k-1}^B, 0.5\}, \rho_{2k-1}^B \leq n$. It thus holds that

$$\begin{aligned}\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] &= \mathbb{E}[(\rho_{2k}^A + U_{2k-1}^B - \min\{\rho_{2k-1}^B, 0.5\}) + (\Psi_k^A - \Psi_k)] \\ &= \mathbb{E}[U_{2k-1}^B - \min\{\rho_{2k-1}^B, 0.5\} + \rho_{2k}^A + (\Psi_k^A - \Psi_k)].\end{aligned}$$

Again, we split the analysis into cases.

Case 1: $(y_{2k-1}^*, x_{2k-1}^*, \delta_{2k-1}^*)$ does not exist.

- $\mathbb{E}[U_{2k-1}^B] = 0$ because Bob does not perform an update.
- $\rho_{2k}^A \geq 0.5$ because Bob's next message is $\mathsf{ECC}(\hat{x}, y, \bullet)$.
- $\mathbb{E}[\Psi_k^A - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k-1}$ since $\psi_{2k-1}^B = \psi_{2k-2}^B < n$ because Bob never updates.
- $\alpha_{2k-1} \geq \left(\frac{1}{6} - \epsilon\right)$ because $(y_{2k-1}^*, x_{2k-1}^*, \delta_{2k-1}^*)$ does not exist.

$$\begin{aligned}\implies \mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] &\geq \mathbb{E}[U_{2k-1}^B] - \min\{\rho_{2k-1}^B, 0.5\} + \mathbb{E}[\rho_{2k}^A] + \mathbb{E}[(\Psi_k^A - \Psi_k)] \\ &\geq 0 - 0.5 + 0.5 + (0.5 - 3\epsilon - 3\alpha_{2k}) \\ &\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

Case 2: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) = (y_{2k-1}^*, x_{2k-1}^*, \delta_{2k-1}^*)$ and ($x_{2k-1} = \hat{x}$ or $\delta_{2k-1} = \bullet$).

- $\mathbb{E}[U_{2k-1}^B] \geq 0.5 - 3\alpha_{2k-1}$. Bob makes a good update with probability $p_{2k-1}^* \geq 0.5 - 3d_{2k-1}^*$ and $\alpha_{2k-1} \geq d_{2k-1}^*$ giving the desired result.

- $\mathbb{E}[\rho^A_{2k} + (\Psi^A_k - \Psi_k)] \geq 1 - 3\epsilon - 3\alpha_{2k}$. To see this, we consider if $\psi^B_{2k-1} \geq n$ or $\psi^B_{2k-1} < n$. If $\psi^B_{2k-1} \geq n$, Bob must've made a good update to message $2k-1$, so it holds that $\rho^A_{2k} = 1$ (since Bob's next message is $\mathsf{ECC}(x, y_{2k-1} = \hat{y}, \mathsf{op}_y(y_{2k-1})))$ and $\mathbb{E}[\Psi^A_k - \Psi_k \mid \psi^B_{2k-1} \geq n] \geq -3\epsilon - 3\alpha_{2k}$, so that $\mathbb{E}[\rho^A_{2k} + (\Psi^A_k - \Psi_k) \mid \psi^B_{2k-1} \geq n] \geq 1 - 3\epsilon - 3\alpha_{2k}$.

  On the other hand, if $\psi^B_{2k-1} < n$, $\rho^A_{2k} \geq 0.5$ since Bob's next message is $\mathsf{ECC}(\hat{x}, y_{2k-1}, \mathsf{op}_y(y_{2k-1}))$, and $\mathbb{E}[\Psi^A_k - \Psi_k \mid \psi^B_{2k-1} < n] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$, so that $\mathbb{E}[\rho^A_{2k} + (\Psi^A_k - \Psi_k) \mid \psi^B_{2k-1} < n] \geq 1 - 3\epsilon - 3\alpha_{2k}$.

$$\begin{aligned}
\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1}] - \min\{\rho^B_{2k-1}, 0.5\} + \mathbb{E}[\rho^A_{2k} + (\Psi^A_k - \Psi_k)] \\
&\geq (0.5 - 3\alpha_{2k-1}) - 0.5 + (1 - 3\epsilon - 3\alpha_{2k}) \\
&= 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.
\end{aligned}$$

Case 3: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) = (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ *and* $(x_{2k-1} \neq \hat{x}$ *and* $\delta_{2k-1} \neq \bullet)$*.*

- $U^B_{2k-1} = 0$ because Bob does not perform an update for the given values of x^*_{2k-1} and δ^*_{2k-1}.
- $\rho^B_{2k-1} = 0$ for the same reason.
- $\rho^A_{2k} \geq 0.5$ because Bob's next message is $\mathsf{ECC}(\hat{x}, y_{2k-1}, \mathsf{op}_y(y_{2k-1}))$.
- $\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$ since $\psi^B_{2k-1} = \psi^B_{2k-2} < n$.

$$\begin{aligned}
\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1}] - \min\{\rho^B_{2k-1}, 0.5\} + \mathbb{E}[\rho^A_{2k}] + \mathbb{E}[(\Psi^A_k - \Psi_k)] \\
&\geq 0 - 0 + 0.5 + (0.5 - 3\epsilon - 3\alpha_{2k}) \\
&\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.
\end{aligned}$$

Case 4: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ *and* $(y^*_{2k-1} = y$ *and* $x^*_{2k-1} = \hat{x})$*.*

Subcase 4.1: The update corresponding to Bob receiving $\mathsf{ECC}(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ *is bad or none.*

- $\mathbb{E}[U^B_{2k-1}] \geq 0.5 - 3\epsilon - 3\alpha_{2k-1}$. Bob makes a bad update with probability at most $p_{2k-1} \leq 1 - 3d^*_{2k-1}$ and $d^*_{2k-1} \geq (\frac{1}{2} - \epsilon) - \alpha_{2k-1}$.
- $\rho^A_{2k} \geq 0.5$ because Bob's response is $\mathsf{ECC}(\hat{x}, y, \bullet)$
- $\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$ because Bob could only have made a bad update, so $\psi^B_{2k-1} < n$.

$$\begin{aligned}
\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1}] - \min\{\rho^B_{2k-1}, 0.5\} + \mathbb{E}[\rho^A_{2k}] + \mathbb{E}[\Psi^A_k - \Psi_k] \\
&\geq (0.5 - 3\epsilon - 3\alpha_{2k-1}) - 0.5 + 0.5 + (0.5 - 3\epsilon - 3\alpha_{2k}) \\
&= 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.
\end{aligned}$$

Subcase 4.2: The update corresponding to Bob receiving $\mathsf{ECC}(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ *is good.*

- $U^B_{2k-1} + \mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$. In the case that $\psi^B_{2k-1} \geq n$, Bob must've made a good update to message $2k-1$, so $U^B_{2k-1} = 1$, and we have that $\mathbb{E}[\Psi^A_k - \Psi_k] \geq -3\epsilon - 3\alpha_{2k}$. In the case that $\psi^B_{2k-1} < n$, we have $U^B_{2k-1} \geq 0$ and $\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$.

- $\rho^A_{2k} \geq 0.5$ because Bob's response is $\mathsf{ECC}(\hat{x}, y, \bullet)$.

- $\alpha_{2k-1} \geq \left(\frac{1}{6} - \epsilon\right)$ because $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ so $\alpha_{2k-1} \geq (\frac{1}{2} - \epsilon) - d^*_{2k-1} \geq \frac{1}{2} - \epsilon - \frac{1}{3} = \frac{1}{6} - \epsilon$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1} + (\Psi^A_k - \Psi_k)] - \min\{\rho^B_{2k-1}, 0.5\} + \mathbb{E}[\rho^A_{2k}] \\ &\geq (0.5 - 3\epsilon - 3\alpha_{2k}) - 0.5 + 0.5 \\ &\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k},\end{aligned}$$

Case 5: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ and ($y^*_{2k-1} \neq y$ or $x^*_{2k-1} \neq \hat{x}$).

Subcase 5.1: The update corresponding to Bob receiving $\mathsf{ECC}(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ is bad or none.

- $\mathbb{E}[U^B_{2k-1}] \geq 1 - 3\epsilon - 3\alpha_{2k-1}$ because Bob makes a bad update with probability at most $p_{2k-1} \leq 0.5 - 3d^*_{2k-1}$ and $d^*_{2k-1} \geq \left(\frac{1}{2} - \epsilon\right) - \alpha_{2k-1}$.

- $\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$ because Bob could only have made a bad update so $\psi^B_{2k-1} < n$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1}] - \min\{\rho^B_{2k-1}, 0.5\} + \mathbb{E}[\rho^A_{2k}] + \mathbb{E}[\Psi^A_k - \Psi_k] \\ &\geq (1 - 3\epsilon - 3\alpha_{2k-1}) - 0.5 + 0 + (0.5 - 3\epsilon - 3\alpha_{2k}) \\ &= 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

Subcase 5.2: The update corresponding to Bob receiving $\mathsf{ECC}(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ is good.

- $U^B_{2k-1} + \mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$. If $U^B_{2k-1} = 1$, then $\mathbb{E}[\Psi^A_k - \Psi_k] \geq -3\epsilon - 3\alpha_{2k}$. If $U^B_{2k-1} = 0$, then $\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$. Either way $U^B_{2k-1} + \mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}$.

- $\alpha_{2k-1} \geq \frac{1}{3}$ because $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ so $\alpha_{2k-1} \geq \left(\frac{1}{2} - \epsilon\right) - d^*_{2k-1} \geq \left(\frac{1}{2} - \epsilon\right) - \left(\frac{1}{6} - \epsilon\right) = \frac{1}{3}$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &\geq \mathbb{E}[U^B_{2k-1} + (\Psi^A_k - \Psi_k)] - \min\{\rho^B_{2k-1}, 0.5\} + \rho^A_{2k} \\ &\geq (0.5 - 3\epsilon - 3\alpha_{2k}) - 0.5 + 0 \\ &\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k},\end{aligned}$$

Proof of Lemma 4.8 when $\psi^B_{2k-2} \geq n$. Finally, we consider the case where $\psi^B_{2k-2} \geq n$. We have that

$$\begin{aligned}\Psi_k - \Psi^A_{k-1} &= \psi^A_{2k-1} + \rho^A_{2k} + \min\{\psi^B_{2k-1}, n\} - \psi^A_{2k-2} - \min\{\psi^B_{2k-2} + \min\{\rho^B_{2k-1}, 0.5\}, n\} \\ &= \rho^A_{2k} + \min\{\psi^B_{2k-1}, n\} - n\end{aligned}$$

since $\psi^A_{2k-1} = \psi^A_{2k-2}$ and $\psi^B_{2k-2} + \min\{\rho^B_{2k-1}, 0.5\} \geq \psi^B_{2k-2} \geq n$. Then

$$\mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] = \mathbb{E}[\rho^A_{2k} + (\min\{\psi^B_{2k-1}, n\} - n) + (\Psi^A_k - \Psi_k)].$$

Again, we split the analysis into cases. Note that $\mathbb{E}[\Psi^A_k - \Psi_k] \geq -3\epsilon - 3\alpha_{2k}$ by Claim 4.9.

Case 1: $(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ does not exist.

- $\rho^A_{2k} \geq 0.5$ because Bob's next message is $\mathsf{ECC}(\hat{x}, y, \bullet)$.
- $\psi^B_{2k-1} \geq n$ still because Bob does not update.
- $\alpha_{2k-1} \geq \left(\frac{1}{6} - \epsilon\right)$ becuase $(y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ does not exist.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &= \mathbb{E}[\rho^A_{2k} + \min\{\psi^B_{2k-1}, n\} - n] + \mathbb{E}[\Psi^A_k - \Psi_k] \\ &\geq 0.5 + n - n + (-3\epsilon - 3\alpha_{2k}) \\ &\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

Case 2: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) = (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$.

- $\psi^B_{2k-1} \geq n$ because Bob cannot perform a bad update.
- $\rho^A_{2k} = 1$ because Bob's next message is $\mathsf{ECC}(\hat{x} = x, y_{2k-1} = \hat{y}, \mathsf{op}_y(y_{2k-1}))$.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &= \mathbb{E}[\rho^A_{2k} + \min\{\psi^B_{2k-1}, n\} - n] + \mathbb{E}[\Psi^A_k - \Psi_k] \\ &\geq 1 + n - n + (-3\epsilon - 3\alpha_{2k}) \\ &\geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

Case 3: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ and ($y^*_{2k-1} = y$ and $x^*_{2k-1} = \hat{x}$).

- $\rho^A_{2k} \geq 0.5$ because Bob's next message to Alice is $\mathsf{ECC}(\hat{x}, y, \bullet)$.
- $\mathbb{E}[\min\{\psi^B_{2k-1}, n\} - n] \geq 0.5 - 3\epsilon - 3\alpha_{2k-1}$. Bob makes a bad update with probability $\leq p^*_{2k-1} = 1 - 3d^*_{2k-1}$. Then $\mathbb{E}[\min\{\psi^B_{2k-1}, n\} - n] \geq 3d^*_{2k-1} - 1$, so using $d^*_{2k-1} \geq \left(\frac{1}{2} - \epsilon\right) - \alpha_{2k-1}$ gives the desired result.

$$\begin{aligned}\implies \mathbb{E}[\Psi^A_k - \Psi^A_{k-1}] &= \mathbb{E}[\rho^A_{2k} + \min\{\psi^B_{2k-1}, n\} - n] + \mathbb{E}[\Psi^A_k - \Psi_k] \\ &\geq 0.5 + (0.5 - 3\epsilon - 3\alpha_{2k-1}) + (-3\epsilon - 3\alpha_{2k}) \\ &= 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

Case 4: $(y_{2k-1}, x_{2k-1}, \delta_{2k-1}) \neq (y^*_{2k-1}, x^*_{2k-1}, \delta^*_{2k-1})$ and ($y^*_{2k-1} \neq y$ or $x^*_{2k-1} \neq \hat{x}$).

- $\rho_{2k}^A \geq 0$ because Alice will never send a message causing Bob to make a bad update.
- $\mathbb{E}[\min\{\psi_{2k-1}^B, n\} - n] \geq 1 - 3\epsilon - 3\alpha_{2k-1}$. Bob makes a bad update with probability $\leq p_{2k-1}^* = 0.5 - 3d_{2k-1}^*$. Using $d_{2k-1}^* \geq \left(\frac{1}{2} - \epsilon\right) - \alpha_{2k-1}$ gives $\mathbb{E}[\min\{\psi_{2k-1}^B, n\} - n] \geq -(0.5 - 3d_{2k-1}^*) \geq 1 - 3\epsilon - 3\alpha_{2k-1}$.

$$\begin{aligned}\implies \mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] &= \mathbb{E}[\rho_{2k}^A] + \mathbb{E}[\min\{\psi_{2k-1}^B, n\} - n] + \mathbb{E}[\Psi_k^A - \Psi_k] \\ &\geq 0 + (1 - 3\epsilon - 3\alpha_{2k-1}) + (-3\epsilon - 3\alpha_{2k}) \\ &= 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}.\end{aligned}$$

5 Binary Interactive Protocol Resilient to $\frac{1}{2}$ Erasures

In this section, we present a non-adaptive interactive protocol where Alice and Bob simulate an existing protocol π_0 via a protocol π resilient to $\frac{1}{2} - \epsilon$ erasures, for any $\epsilon > 0$.

This result is optimal; no protocol is resilient to $\frac{1}{2}$ erasures for all possible functions Alice and Bob might want to compute.

Theorem 5.1 ([FGOS15]). *For all $n > 0$, there exists a function $f(x, y)$ of Alice and Bob's inputs $x, y \in \{0, 1\}^n$, such that any non-adaptive interactive protocol over the binary erasure channel that computes $f(x, y)$ succeeds with probability at most $\frac{1}{2}$ if a $\frac{1}{2}$ fraction of the transmissions are erased.*

In our protocol, Alice and Bob jointly recreate the transcript of π_0 by sending their next message in π_0 until they are sure the other party has received it. How they do this while staying in sync is explained informally in Section 2.2.

5.1 Formal Protocol

Protocol 2 : Protocol Resilient to $\frac{1}{2} - 4\epsilon$ Erasures

Let π_0 be an error-free binary protocol that runs in n_0 rounds. We may assume that π_0 is alternating (i.e. Alice and Bob take turns speaking with Alice speaking first) with at most a factor of 2 blowup in the round complexity. We pad π_0 with 1's so that past round n_0, Alice and Bob both send 1 every round.

Our protocol π occurs in $N = \frac{2n_0}{\epsilon^2}$ rounds, where Alice and Bob alternate speaking with Alice speaking first. These rounds are partitioned into $\frac{n_0}{\epsilon}$ blocks of $\frac{2}{\epsilon}$ rounds, so that in each block Alice and Bob each speak $\frac{1}{\epsilon}$ bits alternatingly.[a]

Alice and Bob each have an internal mode, which is either speaker, listener, or passer. They also track an internal transcript T equal to what they believe the current noiseless protocol is, including the next bit they are trying to send. Thus, initially, $T = \emptyset$ for Bob, and for Alice T is her first message of π. Our protocol begins with Alice in speaker mode and Bob in listener mode.

Alice and Bob stay in the same mode for an entire block, only potentially transitioning at the end of the block. Their behavior in each mode is described as follows.

listener:
- Let $\beta \in \{0, 1, \perp\}$ be the most recently received message from the other party. If $\beta = \perp$, send 0. Otherwise send 1.
- *Transition Condition:* If the first non-erased bit received in the block was 0, switch to

being in speaker mode at the end of the block. Also recall the sequence of bits received in the last block prior to this one where not all incoming messages were erased, and let $b = 0$ if at least one 0 was received and $b = 1$ otherwise. Let b' be the next bit the party would send if the transcript so far is $T||b$, and set $T \leftarrow T||b||b'$.

speaker:
- Let b be the the last bit of the the party's transcript T (b is their next message to send). Send 1 repeatedly until receiving a 1, then send b for the rest of the block.
- *Transition Condition:* If ever 1 was received after having switched to sending b, switch to being in passer mode at the end of the block. Otherwise, remain in speaker mode for the next block.

passer:
- Always send 0.
- *Transition Condition:* If a 1 was received in the block, switch to being in listener mode at the end of the block.

[a]We can assume that $\frac{1}{\epsilon}$ is an integer; otherwise, take a slightly smaller ϵ for which $\frac{1}{\epsilon} \in \mathbb{N}$.

5.2 Analysis

We begin by proving several claims about Protocol 2.

Claim 5.2. *Alice and Bob are never in the same mode at the same time.*

Proof. We show that if Alice and Bob are in two different modes at the beginning of a block, then they cannot be in the same mode at the end of that block. Note that they transition between modes according to the cycle listener $\rightarrow$ speaker $\rightarrow$ passer $\rightarrow \cdots$ and cannot transition more than once per block.

First assume Alice and Bob are listener and speaker in some order at the beginning of a block. It suffices to show the listener cannot become a speaker within that block. This is true because the listener only transitions if the first bit heard within the block is a 0 but the speaker only sends 0 after they receive a 1 confirming that a 1 has been received.

If Alice and Bob are passer and listener in some order, we show that the passer can only become a listener if the listener becomes a speaker. The passer only becomes a listener when they receive a 1, and the listener only sends 1 when they receive a (non-erased) bit from the passer. This bit received from the passer must be a 0, which will cause the listener to transition to speaker mode.

Finally, if Alice and Bob are speaker and passer in some order it suffices to show the speaker cannot become a passer. This is true because the passer only ever sends 0 in the block, and the speaker only transitions if they heard at least two 1's. □

Claim 5.3. *The party that most recently left* listener *mode (or Alice if it is the start of the protocol) has a transcript T that is* 1 *bit longer than the other party's.*

Proof. This is true at the start of the protocol: Alice's T is length 1, and Bob's T is length 0. Since Alice and Bob cycle through listener $\rightarrow$ speaker $\rightarrow$ passer $\rightarrow \cdots$ without ever being in the same mode at the same time by Claim 5.2, they alternate leaving listener mode starting with Bob (who begins in listener mode). Thus, every time a party leaves listener mode they add 2 bits to T, and the claim follows. □

Claim 5.4. *On inputs x, y, a party's simulated transcript T is always a prefix of $\pi_0(x, y)$.*

Proof. A party only modifies T upon exiting listener mode, when they add two bits, one for the other party's message and one for their own. We show that the first bit they added must be the correct next bit of $\pi_0(x, y)$; it then follows that both bits must be the correct next two bits.

In the block B before the party (w.l.o.g. Alice) exits listener mode, the first bit she received in that block must've been a 0. This implies that Bob was in passer mode in block B: he cannot also be in listener mode by Claim 5.2, and he cannot be in speaker mode since then he'd only send 0 *after* receiving a 1 confirming Alice's reception of a 1. The last block R prior to B that Bob was in speaker mode trying to send a bit b, he received a 1 from Alice confirming the reception of the bit b. Then, Alice must've received a nonzero sequence of 1's followed by a nonzero sequence of b's in block R. This must have also been the last block prior to block B that Alice received *any* bits: Bob sent only 0's in blocks $R+1, \ldots, B$, and block B is the first time that Alice received a 0. Thus, Alice determines Bob's bit b correctly and appends it and her next message to her transcript.

□

Claim 5.5. *If a block has at most $\frac{1}{\epsilon} - 3$ corruptions, then at least one of Alice and Bob transitions modes at the end of the block.*

Proof. To show this, we consider the possible combinations of Alice and Bob's starting modes.

If Alice and Bob are in speaker and listener mode, respectively, there are at least 2 (in fact 3) pairs of consecutive Alice-Bob rounds for which neither message is erased, since the adversary can only erase half of the communication for all but 3 Alice-Bob pairs. Let the bit of π_0 Alice is trying to send be b. Then, in the first such pair, Alice sends 1 and receives a 1 from Bob, and in the second such pair, she sends b and receives a 1. Then, at the end of the block, she transitions to passer mode. The case where Alice is listener and Bob is speaker is identical, except we disregard Alice's first and last rounds and consider Bob-Alice pairs of messages. There are still at least 2 non-erased pairs, which is enough for Bob to communicate the two bits $1, b$ and receive confirmation bits.

If Alice and Bob are in passer and listener mode, respectively, then consider Alice-Bob pairs of consecutive rounds. There is at least 1 such pair with no erasures. In this pair, Bob must hear Alice's 0 so that he leaves listener mode at the end of the block. The case where Alice is in listener mode and Bob is in passer mode works analogously by grouping Bob-Alice rounds, ignoring the first and last rounds of the block.

If Alice and Bob are in speaker and passer mode, respectively, Bob only sends 0's so that Alice only sends 1's the entire block. Consider Alice-Bob pairs of consecutive rounds. There is at least 1 such pair with no erasures. In the first such pair, Bob hears Alice's 1 so that he switches to listener mode at the end of the block. The case where Alice is in passer mode and Bob is in speaker mode works analogously, group Bob-Alice rounds and ignoring the first and last rounds of the block. □

Theorem 5.6. *Protocol 2 is resilient to $\frac{1}{2} - 4\epsilon$ fraction of erasures.*

Proof. First we claim that if there are at least $3n_0 + 6$ blocks at the end of which someone switches modes, then each party must leave listener mode at least $\frac{n_0}{2}$ times. To see this, note that at least one party must've switched modes at least $\frac{3n_0}{2} + 3$ times, so that they cycled through all three modes at least $\frac{n_0}{2} + 1$ times. Since Alice and Bob are never in the same mode at the same time, this implies that the other party must've cycled through all three modes at least $\frac{n_0}{2}$ times. In particular, both parties left listener mode at least $\frac{n_0}{2}$ times.

Each time a party leaves listener mode, their transcript increases by length 2, so each party has a final transcript length of at least n_0. By Claim 5.4 this final transcript is correct.

Now suppose that there are $\leq \left(\frac{1}{2} - 4\epsilon\right)$ total erasures. Let $\hat{n}$ denote the number of blocks with at most $\frac{1}{\epsilon} - 3$ erasures. Then $\hat{n}$ satisfies the following inequality double counting the total number of erasures:

$$\begin{aligned}
&\left(\frac{1}{2} - 4\epsilon\right) \cdot \frac{2n_0}{\epsilon^2} \geq \hat{n} \cdot 0 + \left(\frac{n_0}{\epsilon} - \hat{n}\right) \cdot \left(\frac{1}{\epsilon} - 2\right) \\
\implies\ & \frac{\hat{n}}{\epsilon} > \left(\frac{1}{\epsilon} - 2\right) \cdot \hat{n} \geq \frac{6n_0}{\epsilon} \\
\implies\ & \hat{n} > 6n_0 \geq 3n_0 + 3 \cdot 2 = 3n_0 + 6.
\end{aligned}$$

In the last step, we can assume $n_0 \geq 2$ because Alice and Bob talk at least once each in π_0. As such, the number of blocks where someone switches modes is at least $3n_0 + 6$, so Alice and Bob must both have a correct final transcript of length at least n_0 at the end of the protocol.

□

6 Acknowledgments

The authors would like to thank their advisor, Dr. Yael Tauman Kalai (Microsoft Research and MIT). They would like to thank her for helpful discussions, paper edits, as well as introducing them to interactive coding and others who have studied it in the past. They would also like to thank Michael Kural and Naren Manoj for reading the paper and providing comments.

Rachel Yun Zhang is supported by an Akamai Presidential Fellowship.

References

[Ber64] Elwyn R. Berlekamp. Block coding with noiseless feedback. 1964. 2

[Ber68] Elwyn R. Berlekamp. Block coding for the binary symmetric channel with noiseless, delayless feedback. *Error-correcting Codes*, pages 61–88, 1968. 2

[BR11] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 159–166, New York, NY, USA, 2011. Association for Computing Machinery. 1

[EGH16] Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. *IEEE Trans. Inf. Theory*, 62(8):4575–4588, 2016. 1, 2, 8

[EKS20] Klim Efremenko, Gillat Kol, and Raghuvansh R. Saxena. Binary interactive error resilience beyond $1/8$ (or why $(1/2)^3 > 1/8$). In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 470–481, 2020. 1

[FGOS15] Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. *IEEE Transactions on Information Theory*, 61(1):133–145, 2015. 1, 2, 22

[GH17] Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. *SIAM Journal on Computing*, 46:1449–1472, 01 2017. 2

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 181–190, New York, NY, USA, 2000. Association for Computing Machinery. 7

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. 1

[Sch92] L.J. Schulman. Communication on noisy channels: a coding theorem for computation. In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 724–733, 1992. 1

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 747–756, New York, NY, USA, 1993. Association for Computing Machinery. 1

[Sch96] Leonard J Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996. 1

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 1

[SW92] Joel Spencer and Peter Winkler. Three thresholds for a liar. *Combinatorics, Probability and Computing*, 1(1):81–93, 1992. 2

[Zig76] K.Sh. Zigangirov. Number of correctable errors for transmission over a binary symmetrical channel with feedback. *Problems Inform. Transmission*, 12:85–97, 1976. 2