The Optimal Error Resilience of Interactive Communication Over Binary Channels

by

Rachel Yun Zhang

S.B., Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Author .....
Department of Electrical Engineering and Computer Science
January 24, 2023

Certified by .....
Yael Tauman Kalai
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
Vinod Vaikuntanathan
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

The Optimal Error Resilience of Interactive Communication Over Binary Channels

by

Rachel Yun Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on January 24, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science

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 $f(x, y)$.

We study the optimal error resilience of such a protocol in the face of adversarial bit flips 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. Over the binary alphabet, there has remained a substantial gap in error resilience between the best protocol construction and the best known upper bound, for both bit flips and erasures.

In this thesis, we construct protocols meeting the known upper bounds for both types of errors, thereby closing this gap and resolving the question of optimal error resilience for binary channels. Our schemes for both types of errors have positive rate and are computationally efficient.

Thesis Supervisor: Yael Tauman Kalai
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Vinod Vaikuntanathan
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my wonderful advisors, Yael Tauman Kalai and Vinod Vaikuntanathan, for their guidance and support throughout my time at MIT. They have been so supportful of both my academic and non-academic pursuits, for which I am deeply grateful.

I would especially like to thank Yael Tauman Kalai, who took me on for a UROP when I was an undergraduate student, and invited me to think about some of the most interesting problems in cryptography. It was due to our time together that I decided to pursue a PhD in theoretical computer science.

The work presented in this thesis is joint work with Meghal Gupta, who has been an amazing friend and collaborator. I could not ask for a better partner in crime.

Contents

Chapter 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 over the binary channel with constant information rate[1] that is resilient to bit flip errors in $\frac{1}{240}$[2] of the bits of the protocol. This work begs the natural question: *what is the maximum error resilience possible?* This is precisely the focus of this thesis.

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, for *large constant-sized alphabets*, there are known protocols that achieve optimal error resilience. 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

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

[2] 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 $r - \epsilon$.

best known impossibility bound is $\frac{1}{6}$ [EGH16],[3] 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 thesis, 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.*

1.1 Our Results

1.1.1 Bit Flip Errors

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

As a precursor to our main result, we consider the first the task of *message exchange*, where the goal is for Alice and Bob to learn each other's secret inputs. In this setting, we construct a protocol resilient to the optimal fraction of errors.

Theorem 1.1. *For any $\epsilon > 0$ and 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 $C(\epsilon)$.*

Note that the communication and computational complexity of the above protocol is polynomial in the size of Alice and Bob's inputs, rather than on the minimal size of a noiseless 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. In the next result, we remove this overhead.

Theorem 1.2. *For any $\epsilon > 0$ and any interactive binary protocol π_0 computing a function $f(x, y)$ of Alice and Bob's private inputs x, y, there exists a non-adaptive interactive binary protocol π computing $f(x, y)$ that is resilient to $\frac{1}{6} - \epsilon$ adversarial erasures. The communication complexity is $O_\epsilon(|\pi_0|)$ and the computational complexity is $\tilde{O}_\epsilon(|\pi_0|)$.*

Discussion. Until our results, 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

[3] Technically, the upper bound of $\frac{1}{6}$ was previously proved in the setting that Alice and Bob are both deterministic. In Section 2.2.1, we reprove this upper bound in the case that Alice and Bob are randomized.

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!

1.1.2 Erasures

Over the binary erasure channel, we obtain a positive rate and computationally efficient scheme achieving the optimal $\frac{1}{2}$ erasure resilience:

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 $\frac{1}{6}$.

1.2 Outline

The rest of the thesis is structured as follows.

- In **Chapter 2**, we formally define the task of interactive coding, and discuss knownn upper (impossibility) bounds on the noise resilience for binary schemes.
- Next, in Chapters 3-8, we build our interactive coding scheme for bit flip errors.
  - We begin in **Chapter 3** by constructing an optimally error-resilient coding scheme for the task of message exchange. The design and analysis will encompass many of the key new ideas for obtaining error resilience $\frac{1}{6}$.
  - Next, in **Chapter 4**, we discuss and motivate at a high level the changes necessary to obtain both positive information rate and computational efficiency.
  - The key components necessary to implement the above strategy, *boosting* and *layered codes*, will be introduced in **Chapters 5 and 6**.
  - Finally, in **Chapter 7**, we put together all these tools to obtain our efficient coding scheme with optimal error resilience for bit flip errors.
- Moving ahead, in **Chapter 8**, we demonstrate a scheme that is resilient to the optimal fraction of *erasures*.
- Lastly, in **Chapter 9**, we discuss future directions of work.

Chapter 2

The Landscape

In this chapter, we formally define the model of *non-interactive* coding schemes, and then give upper bounds on the best possible error/erasure resilience of interactive coding schemes.

2.1 Non-Adaptive Coding Schemes: The Model

In this thesis, we will be working with *non-adaptive* coding schemes, that is, where the length of the protocol and speaking order is fixed a-priori. In a sense, non-adaptive protocols with error resilience are the natural interactive generalization of error correcting codes. We formally define this concept below.

Definition 2.1 (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 transmissions, denoted $|\pi|$. In each transmission, a single party fixed beforehand sends a single bit to the other party. At the end of the protocol, each party outputs a guess $\in \{0,1\}^o$.*

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

We are interested in the maximal possible value of α possible in both the case of bit flip errors and erasures.

2.2 Upper Bounds on the Noise Resilience

2.2.1 Upper Bound for Bit Flip Errors

An upper bound of $\frac{1}{6}$ on the optimal error resilience is well known. However, the existing proofs [EGH16, Gel17, SW92] show the upper bound only in the case that Alice and Bob's strategies are deterministic. The proof strategy is essentially the same when Alice and Bob are allowed access to private randomness, but we recreate the proof below for completeness. Our proof is adapted from [Gel17].

Theorem 2.2. *For large enough n, any non-adaptive interactive protocol π over the binary bit flip channel where Alice and Bob are randomized Turing machines that exchange inputs $x, y \in \{0,1\}^n$, succeeds with probability at most $\frac{1}{3}$ if a $\frac{1}{6}$ fraction of the transmissions are corrupted.*

Proof. Without loss of generality, Alice is the party that speaks less during the protocol. Specifically, let Alice speak in $n_A < |\pi|/2$ rounds.

Consider three possible versions of Alice: Alice_a, Alice_b, Alice_c that hold one of three possible inputs a, b, c. Let Alice_a have randomness r_a, Alice_b have randomness r_b and Alice_c have randomness r_c. We demonstrate an attack where for a fixed input y and randomness s that Bob has, he outputs a guess for Alice's input incorrectly when playing with at least one of Alice_a, Alice_b and Alice_c. Then, averaging over all choices of Alice's randomness, Bob's overall probability of outputting an incorrect guess must be at least $\frac{1}{3}$ in one of the cases that Alice has a, b, or c. Eve can employ this attack by simulating the other two versions of Alice herself (the attack does not require knowing Alice's randomness beforehand).

Denote the bit sent by Alice_a with randomness r_a at round t by $b_a(t)$ and define $b_b(t), b_c(1)$ similarly. Eve's attack changes the bit Alice sends at round $t = 1$ to $\text{maj}(b_a(1), b_b(1), b_c(1))$, where $\text{maj}(b_1, b_2, b_3)$ denotes the majority bit out of $\{b_1, b_2, b_3\}$. Then, for at least two inputs, the channel does nothing, and for the third input, the channel may need to flip the bit. At every other round $t < R$ (we set R shortly), she does the same attack (changing Alice's transmission to $\text{maj}(b_a(t), b_b(t), b_c(t))$. Because Bob has the same input and randomness, and receives the same transmission in all cases, his messages in between are identical in all cases.

Denote by $N_a(t)$ the corruption the preceding attack requires through round t if for Alice_a (similarly define $N_b(t), N_c(t)$). Set R to be the minimal round so that second-largest value out of $N_a(R), N_b(R), N_c(R))$ equals $n_A/3$. Without loss of generality, assume c maximizes this value at round R and that b is the second-largest value, that is, $N_b(R) = n_A/3, N_a(R) < n_A/3$. Also note that since the attack makes a single corruption in at most one input every round, we have $R \geq N_a(R) + N_b(R) + N_c(R)$. Finally, note that up to round R, Bob sees exactly the same view whether Alice holds a, b, c. From this point on, we don't care about the input c, as we will show the attack succeeds on inputs a and b.

From Alice's round $R + 1$ until the end of the protocol, if Alice holds a, Eve changes all Alice's transmissions to be what Alice would have sent given that she had the input b. If Alice holds b, the channel does nothing. In Bob's view, exactly the same bits are received between round R and the end of the protocol (and therefore, throughout the entire protocol) whether Alice holds a or b.

We are left to show that the total noise rate is at most $\frac{1}{3}$. If Alice holds b, then the attack stops at round R, when $N_b(R) = n_A/3$, as needed. When Alice holds a, the channel corrupts $N_a(R)$ bits until round R and at most $(n_A - R)$ bits afterwords. Recall that $\text{R} \geq N_a(R) + N_b(R) + N_c(R)$ and that $N_b(R), N_c(R) \geq n_A/3$; thus the attack makes at most

$$N_a(R) + (n_A - R) \leq N_a(R) + n_A - N_a(R) - 2n_A/3 \leq n_A/3$$

corruptions, as needed. Since Bob sees the same view in both cases, he outputs incorrectly for at least one of them. □

2.2.2 Upper Bound for Erasures

For erasures, the upper bound of $\frac{1}{2}$ is fairly straightforward. The proof can be summarized as follows: pick the party that speaks less, and erase everything they say. This only requires $\frac{1}{2}$ erasures, and silences one of the parties for good.

Theorem 2.3 ([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.*

2.3 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 $x \in \{0,1\}^*$.
- $x[i : j]$ denotes the $i \ldots j$'th bits of $x \in \{0,1\}^*$.
- $x||y$ denotes the string x concatenated with the string y.

Chapter 3

Warm-Up: Optimally Error Resilient Message Exchange for Bit Flip Errors

In this chapter, we give our construction of a coding scheme resilient to the optimal $\frac{1}{6}$ fraction of bit flip errors, for the task of message exchange. Recall that the task of message exchange is as follows: Alice and Bob each have a private input x and y respectively, and their goal is to learn the other parties' private input by the end of the protocol. The goal is to ensure that both learn the other's private input even if $\frac{1}{6} - \epsilon$ of the communication is corrupted.

We begin by motivating and discussing our construction at a high level in Section 3.1. Then, in Sections 3.2 and 3.3, we give our construction. Finally, in Section 3.4, we analyze our protocol and show that it is indeed resilient to $\frac{1}{6}$ errors.

3.1 Protocol Overview

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 $\mathsf{ECC}(x^*, y^*, \delta^*)$[1] 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.

However, using this approach is not so simple. When one party asks a question, the other party's response only achieves distance $\frac{2}{3}$ because there are only 4 options for the message, and so we can encode them in a $\frac{2}{3}$ distance ECC. When the party must also send a question, the number of possible messages becomes dependent on n. The next few paragraphs show how we achieve distance $\frac{2}{3}$ between the relevant options even when the number of possible messages is large, and why our solution works.

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 3.4.) 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 is still not clear why this fixes our problem: 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

[1] 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.

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. ≪Meghal: next revision I think we should move the our protocol section to after discussing partial corruptions.≫ 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 (note that there are 4 such codewords), 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 $\mathsf{ECC}(\hat{y}, x, \bullet)$.

In the rest of this chapter, we formalize the above discussion.

3.2 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 3.1 ($(z, c) \oplus \delta$)**.** *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 3.2 (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

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

throughout the section.

Definition 3.3 (ECC)**.** *For a given $\epsilon > 0$, we define the error correcting code family*

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

with the following properties:

- $M = O_\epsilon(n)$.
- *For any $n \in \mathbb{N}$ and for any $(z_0, z_0') \neq (z_1, z_1') \in \{0,1\}^{\leq n} \times \{0,1\}^{\leq n}$ and $\delta_0, \delta_1 \in \{0, 1, \leftarrow, \bullet\}$,*

$$\Delta\big(\mathsf{ECC}_{\epsilon,n}(z_0, z_0', \delta_0), \mathsf{ECC}_{\epsilon,n}(z_1, z_1', \delta_1)\big) \geq \left(\frac{1}{2} - \epsilon\right) \cdot M, \tag{3.1}$$

- *For any $n \in \mathbb{N}$ and for any $z, z' \in \{0,1\}^{\leq n} \times \{0,1\}^{\leq n}$ and $\delta_0 \neq \delta_1 \in \{0, 1, \leftarrow, \bullet\}$,*

$$\Delta\big(\mathsf{ECC}_{\epsilon,n}(z, z', \delta_0), \mathsf{ECC}_{\epsilon,n}(z, z', \delta_1)\big) \geq \frac{2}{3}M. \tag{3.2}$$

- *Decoding up to $\frac{1}{6} - \epsilon$ errors can be done in time $C(\epsilon)n^{O(1)}$ for some constant $C(\epsilon)$.*

Claim 3.4. *For all $\epsilon > 0$, an explicit error correcting code family ECC_ϵ from Definition 3.3 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 $\mathsf{ECC}' : \{0,1\}^{\leq n} \times \{0,1\}^{\leq n} \to \{0,1\}^{M/3}$ 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 $\leq \frac{1}{4} - \epsilon$ errors. This exists for some $M = O_\epsilon(n)$ by Theorem **??**. 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 (7.1) 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 $\geq \frac{1}{2} - \epsilon$, and Equation (7.2) 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 $\leq \left(\frac{1}{6} - \epsilon\right)$ 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:

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

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

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

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 $d_\delta \leq \left(\frac{1}{6} - \epsilon\right)$, 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 $\leq \frac{1}{6} - \epsilon$ 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 $\leq \frac{1}{6} - \epsilon$ to a codeword $\mathsf{ECC}_\epsilon(z, z', \delta')$, then $S_{\delta'}$ is relative distance $\leq \frac{1}{6} - \epsilon$ 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 chapter, ECC will denote the error correcting code $\mathsf{ECC}_{\epsilon,n}$ as defined in Definition 3.3 when n and ϵ are clear.

3.3 Formal Protocol

We are now ready to formally state the protocol.

Protocol 3-1 : 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 $1, \ldots, T$, 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 $\hat{y} \in \{0,1\}^{\leq n}$ and $\hat{x} \in \{0,1\}^{\leq n}$ 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.

$\boldsymbol{f}$ 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 $\mathsf{ECC}(\hat{y}, x, \bullet)$.

3.4 Analysis

We now state our main error resilience theorem.

Theorem 3.5. *Protocol 7-1 is resilient to a* $\left(\frac{1}{6} - 2\epsilon\right)$*-fraction of errors with probability* $1 - 2\exp\left(\frac{-\epsilon n}{100}\right)$.[2]

To prove Theorem 7.6, 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 $0, \ldots, t$.

[2] 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 3.6. *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 $1, \ldots, t-1$, 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 $1, \ldots, t-1$, 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 3.7. *If an α_{2k-1} fraction of message $2k-1$ and an α_{2k} fraction of message $2k$ is corrupted, then*

$$\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \geq 1 - 6\epsilon - 3\alpha_{2k-1} - 3\alpha_{2k}. \tag{3.3}$$

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

$$\mathbb{E}[\Psi_k^B - \Psi_{k-1}^B] \geq 1 - 6\epsilon - 3\alpha_{2k-2} - 3\alpha_{2k-1}. \tag{3.4}$$

We delay the proof of Lemma 3.7 to Section 3.4.1 as it is quite long and instead use it to complete the proof of Theorem 7.6.

Proof of Theorem 7.6. Consider any adversarial corruption of Protocol 7-1 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 $\alpha_1, \ldots, \alpha_T$ denote the fractional number of corruptions in messages $1, \ldots, T$, such that $\alpha_1 + \cdots + \alpha_T < \left(\frac{1}{6} - 2\epsilon\right) \cdot T$. 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 3.7,

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

so Φ_k^A is a submartingale with bounded distance

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

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 7.10 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

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

□

3.4.1 Proof of Lemma 3.7

We only prove Inequality (3.3), as the proof for Inequality (3.4) 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 3.8. *If $\psi^B_{2k-1} < n$, then*

$$\mathbb{E}[\Psi^A_k - \Psi_k] \geq 0.5 - 3\epsilon - 3\alpha_{2k}. \tag{3.5}$$

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

$$\mathbb{E}[\Psi^A - \Psi_k] \geq -3\epsilon - 3\alpha_{2k}. \tag{3.6}$$

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 (3.5) and (3.6), we will show they hold for any value of $(x_{2k}, y_{2k}, \delta_{2k})$, which implies they hold in expectation.

Proof of Equation (3.5)**.** 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 7.10, 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 (7.1) 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 (3.6)**.** 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 7.10, $x_{2k} = x$. 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 $\alpha_{2k} \geq d_{2k}^*$.

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

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

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.

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

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

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

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

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

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

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

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

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

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

□

We return now to the proof of Lemma 3.7. 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 3.8 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 (3.3) and (3.4) 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 3.8, 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 3.7 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 3.7 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 3.8.

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 $x^*_{2k-1} \neq \hat{x})$.

- $\rho^A_{2k} \geq 0$ because Alice will never send a message causing Bob to make a bad update.
- $\mathbb{E}[\min\{\psi^B_{2k-1}, n\} - n] \geq 1 - 3\epsilon - 3\alpha_{2k-1}$. 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 $\mathbb{E}[\min\{\psi^B_{2k-1}, n\} - n] \geq -(0.5 - 3d^*_{2k-1}) \geq 1 - 3\epsilon - 3\alpha_{2k-1}$.

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

Chapter 4

The Road to an Efficient Scheme for Bit Flip Errors

Proceeding from our $\frac{1}{6}$-error resilient scheme for message exchange, in this section, we outline at a high level the steps necessary to achieve positive communication rate and computational efficiency.

We begin by recalling at a high level the protocol of Chapter 3, which achieves optimal error resilience $\frac{1}{6} - \epsilon$, but whose communication complexity is quadratic in the input lengths.

Suppose Alice and Bob have private inputs $x, y \in \{0,1\}^n$. Consider the task of *message exchange*, where the goal is for Bob to learn x and for Alice to learn y. The protocol of [GZ22] is a $(\frac{1}{6} - \epsilon)$-error resilient protocol achieving message exchange, where the communication complexity is $O_\epsilon(n^2)$.

The protocol works as follows. Alice and Bob each keep a track of a guess $\hat{y}$ or $\hat{x}$ for the other party's input, initially set to $\emptyset$, and a weight w_A or w_B indicating their confidence for their guess $\hat{y}$ or $\hat{x}$ respectively, initially set to 0.

The idea is that Alice can ask a *question* by sending Bob her guess $\hat{y}$ encoded in an error correcting code. Bob can then send her an *answer* telling her how to update $\hat{y}$ to bring it closer to his actual input y: append 0 (0), append 1 (1), delete the last bit ($\leftarrow$), or "bingo – you got it right!" ($*$). (This last instruction $*$ tells Alice to increase w_A. If Alice receives an instruction to modify $\hat{y}$ while $w_A > 0$, she decreases w_A by 1 instead.) Since Bob's answer is always one of four options, his possible answers can be made to be relative distance $\frac{2}{3}$ apart (e.g. $000, 011, 101, 110$), so that the adversary would have to corrupt $\geq \frac{1}{3}$ of Bob's bits sent (or $\frac{1}{6}$ overall) to prevent Alice from making good updates to $\hat{y}$ (i.e. updates that get $\hat{y}$ closer to y).

Now, since both Alice and Bob have to learn the other's input, Alice and Bob *simultaneously* ask a question and answer the other party's last question. In other words, Alice's message is always of the form $\mathsf{ECC}(\hat{y}, x^*, \delta)$, where x^* is the question she just heard from Bob and δ is the instruction on how to update x^* to bring it closer to x. Similarly, Bob's message is always of the form $\mathsf{ECC}(\hat{x}, y^*, \delta)$. Here, ECC is a code with certain distance properties, including that for any x', y' the four codewords $\{\mathsf{ECC}(x', y', 0), \mathsf{ECC}(x', y', 1), \mathsf{ECC}(x', y', \leftarrow), \mathsf{ECC}(x', y', *)\}$ should be pairwise relative distance $\frac{2}{3}$ from each other.

However, there are two problems with this current algorithm:

(a) The adversary can simultaneously corrupt both the question and answer in Bob's message $\mathsf{ECC}(\hat{x}, \hat{y}, \delta)$ by only corrupting $\frac{1}{2}$ of the message, so that Alice receives an incorrect answer

and thus makes a bad update for only $\frac{1}{2}$ cost.

(b) The adversary can partially corrupt Bob's message (so that the message Alice receives is not any codeword), so Alice does not know what question to answer.

The algorithm of [GZ22] fixes these problems with two additional rules.

- When Alice receives a message $\mathsf{ECC}(x', \hat{y}, \delta')$, she usually only updates with probability 0.5. However, if $x' = x$ (i.e. Bob has already figured out her input), she updates with probability 1.
- When Alice receives a partially corrupted message where she cannot determine what question to answer, she defaults to sending $\mathsf{ECC}(\hat{y}, x, *)$. Correspondingly, when Bob receives any message $\mathsf{ECC}(y', x', *)$ where the update instruction is $*$, he updates $\hat{x}$ to be closer to x'.

Both these new rules require one important fact: that Alice knows what Bob's correct output ought to be (her input x). For us, we will be simulating a noiseless protocol π_0 where the final transcript depends on both parties' private inputs, so that neither Alice nor Bob knows what the correct final transcript ought to be. This is the main barrier to making the protocol of [GZ22] run in time $O_\epsilon(|\pi_0|^2)$ as opposed to in time $O_\epsilon(n^2)$.

4.1 Obtaining Communication Complexity $O_\epsilon(|\pi_0|^2)$

The first modification we will make is to create an interactive coding scheme that can simulate general protocols, instead of just message exchange, in quadratic time. By doing this, we will obtain a protocol with communication complexity $O_\epsilon(|\pi_0|^2)$ instead of $O_\epsilon(n^2)$.

At a high level, in our protocol, in each message Alice and Bob either asks a question *or* answers a received question, *but not both.* This is as opposed to the protocol of [GZ22], in which question asking and answering are always done simultaneously. We remark that this removes issue (a) with the [GZ22] protocol, since now answers no longer have a question component so that all possible answers $\{\mathsf{ECC}(r^*, 0), \mathsf{ECC}(r^*, 1), \mathsf{ECC}(r^*, \leftarrow), \mathsf{ECC}(r^*, \bullet)\}$ to the same question r^* are distance $\frac{2}{3}$ apart.

More concretely, Alice and Bob each keep track of a guess for the complete noiseless transcript, denoted T_A or T_B respectively, along with a weight w_A or w_B signaling how confident they are that the current transcript guess is correct. We have that $w = 0$ unless the corresponding transcript guess T is complete, meaning $|T| = |\pi_0|$. Alice's transcript guess T_A always has odd length, i.e. she is the last to speak, unless T_A is a complete transcript or is the empty transcript. Similarly, Bob's transcript guess T_B always has even length. Let $\mathcal{T}$ denote the noiseless transcript, so that the goal is for Alice and Bob to have $T_A = T_B = \mathcal{T}$ by the end of the protocol. In what follows, we describe the protocol from Alice's point of view, but Bob's behavior is equivalent.

Every round, Alice sends a message of the form $\mathsf{ECC}(T, \delta \in \{0, 1, \leftarrow, ?\})$, where $\delta =?$ signals that she is asking a question and $\delta \in \{0, 1, \leftarrow\}$ signals that she is answering a question. Specifically, when Alice asks a question, she sends $\mathsf{ECC}(T_A, ?)$. She answers a question T_B^* by sending $\mathsf{ECC}(T_B^*, \delta)$, where $\delta \in \{0, 1, \leftarrow\}$ is

- $\leftarrow$ if T_B^* is not consistent with her own behavior on input x.

- her next message 0 or 1 given the consistent transcript prefix T_B^* (if T_B^* is a complete transcript, then her next message is just 1).

Here, ECC is a code satisfying that for any T^* the four words $\mathsf{ECC}(T^*, 0)$, $\mathsf{ECC}(T^*, 1)$, $\mathsf{ECC}(T^*, \leftarrow)$, $\mathsf{ECC}(T^*, ?)$ have relative distance $\frac{2}{3}$ and all other pairs of codewords are relative distance $\frac{1}{2}$ apart. Such a code was shown to exist in [GZ22].

Alice determines whether to ask or answer based on the message she just received:

- As long as she receives an answer (not necessarily to the question she previously asked), she asks a question.
- Whenever Alice receives a question, she answers it. There is an exception, which is when the question received is a complete transcript consistent with Alice's own input x. In this case, Alice asks her own question. This mechanism allows Alice and Bob to switch who is asking vs. answering once the asking party has made sufficient progress and now knows $\mathcal{T}$.

Furthermore, every time Alice receives a message from Bob, she needs to update (T_A, w_A) accordingly:

- When she receives an answer to her question $\mathsf{ECC}(T_A, \delta \in \{0, 1\})$, she concatenates δ and her resulting next message to the end of T_A. (If T_A is a complete transcript, she instead increments w_A.)
- If she receives $\mathsf{ECC}(T_A, \leftarrow)$, assuming $w_A = 0$ she deletes the last two messages (one of hers and one of Bob's) from T_A, and otherwise if $w_A > 0$ she simply decreases w_A by 1.
- If she receives a question $\mathsf{ECC}(T_B^*, ?)$ from Bob, where T_B^* corresponds to a complete transcript that is consistent with her input x, she updates T_A to be one step closer to T_B^* with 0.5 probability.

  There is an exception to this rule, which is when $T_B^* = T_A$. This can only happen if $T_B^* = T_A$ is either $\emptyset$ or a complete transcript, as in general T_A is of odd length and T_B is of even. In this case, with probability 1 instead of 0.5, Alice increases her weight w_A on the transcript T_A by 1. This is because when $T_A = T_B = \mathcal{T}$, we want both Alice and Bob to make more progress simultaneously.[1] Similarly, Bob also needs to be updating with probability 1 whenever he receives a question from Alice equal to T_B.
- Otherwise, she does not update T_A or w_A.

So far, we have described the protocol when the parties receive full codewords. When messages are *partially corrupted* so that the received message is not a codeword, a party will default to asking a question with probability proportional to the distance from the nearest codeword, and otherwise employ the above behavior. This addresses issue (b). We remark that the default message being a question is the second idea that allows us to escape from needing for Alice and Bob to know what the other party's output ought to be, since instead of defaulting to sending the answer $(x, *)$ or $(y, *)$ one now defaults to asking a question.

[1] The potential function we care about is [Alice's progress] + min{[Bob's progress], $|\pi_0|$}, so once Bob's progress is $\geq |\pi_0|$ signaling that $T_B = \mathcal{T}$, we need Alice to be updating with probability 1 each time she correctly receives Bob's message.

4.2 Reducing the Communication Complexity to $O_\epsilon(|\pi_0|)$

Now that we have an optimally error resilient interactive coding scheme that can simulate protocols with $O_\epsilon(|\pi_0|^2)$ communication complexity, the next step is to reduce the communication complexity to $O_\epsilon(|\pi_0|)$.

Currently, the quadratic factor in the communication complexity arises because we need $O_\epsilon(|\pi_0|)$ rounds to simulate the protocol, and in each round the parties are sending either their transcript guess or the transcript guess they are answering, both of which takes $O_\epsilon(|\pi_0|)$ bits. If we could reduce the amount of communication needed to send a transcript guess to $O_\epsilon(1)$, then we could achieve our desired $O_\epsilon(|\pi_0|)$ total communication.

Consider first the task of a party sending their own transcript guess as a question such that each message is only $O_\epsilon(1)$ bits. The traditional solution for this problem in interactive coding is to use *tree codes* [Sch93, Sch96], which are essentially error correcting codes that one can update in an online way. In our setting, since a new transcript guess is a two-bit modification of the last transcript guess, we can have Alice and Bob track a sequence of updates $U_A, U_B \in \{0, 1, \leftarrow, \bullet\}^*$ they have made to obtain their current transcript guess, where $\bullet$ is a placeholder update that simply means "do nothing." Then, the question asker will send just the next two symbols of a tree code encoding of U_A or U_B, which will take $O_\epsilon(1)$ bits per round. The receiver can then decode the entire history of received messages to determine the sequence of updates, which will allow them to determine the transcript being asked.

In our $O_\epsilon(|\pi_0|^2)$ protocol, we had the property that for Alice to successfully decode the asked transcript, she only needed to receive the last message (which contained the entire asked transcript) correctly. However, in a traditional tree code, even if Alice received the last message correctly, she cannot decode the message history if she received a high fraction (specifically more than half) of the previous messages incorrectly. In this paper, we present a new notion of *sensitive* tree codes that in fact satisfy a stronger property, that for all but $\epsilon|w|$ indices i where $w[i] = LTC(x)[i]$, it in fact holds that decoding $w[1:i]$ will *uniquely* give $x[1:i]$. This essentially means that Alice only needs to receive the previous symbol of a sensitive tree code correctly to determine the entire message so far.[2]

Our notion of sensitive tree codes follows a similar construction as *list tree codes*, introduced by Braverman and Efremenko [BE14]. These are codes which guarantee that there is on average some constant number of ways to decode a random prefix of a string w. What we show is that this constant can actually be made 1.

Still, we need answers to have message size $O_\epsilon(1)$ as well. To achieve this, we make the following modification to the answer format. Instead of sending $\mathsf{ECC}(T^*, \delta)$, which has size $O_\epsilon(|\pi_0|)$, a party who wishes to answer the transcript specified by the sequence of operations U^* instead sends $\mathsf{ECC}(\sigma, \delta)$, where σ is the last two symbols in the list tree code encoding of $(U^*||\bullet\bullet)$.

There is still one case where the new protocol is not analogous to the one from Section 4.1. In the protocol from Section 4.1, when Alice is asking the same transcript T' that she is answering, she sends $\mathsf{ECC}(T', ?)$ as a question. Bob will notice that T' happens to be the same as the question he asked, and update with probability 1. In some sense, this message gives Alice the benefits of both asking and answering a question. However, in the new setup, in order to ask a question, Alice has

[2]Sensitive tree codes can also be thought of as codes where the message can (usually) be decoded uniquely as long as the suffix distance to the original codeword is at most $1 - \epsilon$. Previous results only guaranteed a message could be decoded correctly when the suffix distance was $\frac{1}{2} - \epsilon$ to the original codeword; for example Lemma 2.3 in [Gel17].

to send the last two symbols of the encoding of U_A, but in order to answer U_B^* she has to send the last two symbols of U_B^*. The issue is that these symbols may not be the same, even if U_A and U_B^* correspond to the same complete transcript T'.

This leads us to define a new sort of online-updatable code, where if two histories correspond to the same transcript, even if the histories themselves are different, the next tree code encoding of a given edge is the same. This requires defining a code on a particular graph rather than on trees.

4.3 Codes on Graphs

Consider the rooted $|\Sigma_{in}|$-ary tree $\mathcal{C}$. A sequence of symbols $\in \Sigma_{in}$ can be associated with a rooted path of $\mathcal{C}$ in the natural way. A sensitive tree code is then an assignment of symbols in Σ_{out} to the edges of $\mathcal{C}$. To encode a string $x \in \Sigma_{in}^k$, one simply traverses the corresponding rooted path and writes down the symbols seen. This gives an encoding $\in \Sigma_{out}^k$.

The problem with using sensitive tree codes for our purposes is that Alice may have followed one path to get to the correct transcript $T_A = \mathcal{T}$ while Bob followed another to get to $T_B = \mathcal{T}$. Then, the next edge for Alice is different then the next edge for Bob, which means that one cannot hope to coincide sending the next symbol of one's own tree code with answering the other's.

Our key observation is that the encoding of the next symbol depends only on the transcript so far, not the full history of symbols. So, we can actually coincide all nodes of $\mathcal{C}$ that lead to the same transcript. We define the following graph.

The Graph. The graph G that we will be interested in is defined as follows:

- G is a directed graph with vertices partitioned into layers $1, 2, \ldots$. In the i'th layer, there is a vertex for each possible transcripts of length $\leq i$. In particular, there is one vertex in the 0'th layer, namely, the empty string.
- We set $\Sigma_{in} = \{0, 1, \leftarrow, \bullet\}$ to be the possible update instructions, where $\bullet$ means simply "do nothing." Each vertex in the i'th layer has 4 children in the $(i+1)$'th layer, corresponding to the 4 resulting transcripts obtained by applying an instruction in Σ_{in} to the vertex's associated transcript.

Note that any sequence of updates $\in (\Sigma_{in})^*$ corresponds to a rooted path in G. Furthermore, any two equal length sequences of updates that result in the same transcript end at the same node.

The Code on G. We define a *layered code* to be an assignment of elements of Σ_{out} to the edges of G. Then, to encode $x \in (\Sigma_{in})^*$, one simply follows the path specified by x and records the $|x|$ symbols seen on the edges.

We will use a specific layered code C that exhibits the same behavior as the sensitive tree codes we defined in Section 4.2. We call these codes *sensitive layered codes*. In particular, the property we want is that for all but $\epsilon|w|$ indices i where $w[i] = \mathsf{C}(x)[i]$, decoding $w[1:i]$ gives a unique *vertex* (i.e. transcript guess) equal to the vertex at the end of the rooted path specified by $x[1:i]$.

We will not go into depth how such to prove the existence of such a code here, but instead refer the reader to Section **??** for a comprehensive discussion. While much of our construction and proofs are

motivated by the list tree codes of [BE14], we remark that there are several subtleties that need to be carefully addressed.

4.4 Boosting to Achieve Computational Efficiency

Thus far, we have described how to obtain an interactive coding scheme that is resilient to $\frac{1}{6}-\epsilon$ error and has communication complexity linear in the size of the original protocol. Unfortunately, since decoding our sensitive layered code is inefficient (in fact, takes exponential time), this means that the computation needed by both parties is exponential in $|\pi_0|$. Thus, the final needed component is a way to make our scheme efficiently computable.

Over a large alphabet, an efficiently computable, positive rate scheme that is maximally error resilient was constructed by [GH13]. They obtained this efficient scheme in two steps: first by *boosting* a known inefficient, exponential-time scheme [BR11] to obtain an efficient protocol with a list-decoding guarantee, and second by applying a transformation that takes a list-decoding protocol to a unique-decoding protocol. We remark that this second transformation crucially relies on using a large alphabet and thus will not be permittable for us.

The boosted list-protocol is obtained as follows. First, they split up their original noiseless protocol into $\log^4|\pi_0|$ size chunks. Then, they use their inefficient scheme to simulate the following noiseless subprotocol $O_\epsilon(\frac{|\pi_0|}{\log^4|\pi_0|})$ times:

- Alice and Bob first find the longest transcript they have both simulated so far. This takes $O(\log^4|\pi_0|)$ rounds.
- Next, they run the next chunk of $\log^4|\pi_0|$ rounds of the noiseless protocol.

Whenever a simulated subprotocol results in a completed transcript, that complete transcript obtains a vote. At the end, they show that as long as there was not too much corruption, the correct transcript must be one of the transcripts with the most votes (i.e. each party obtains a list of possible transcripts containing the correct one). Note that this results in a protocol with computational complexity $O_\epsilon(\frac{|\pi_0|}{\log^4|\pi_0|})\cdot\exp(\log^4|\pi_0|) = \exp(\text{polylog}|\pi_0|)$ time, which is considerably better than $\exp(|\pi_0|)$. Recursively boosting a second time gets the computational complexity down to $\text{poly}(|\pi_0|)$. A third time reduces the computational complexity to $\tilde{O}_\epsilon(|\pi_0|)$.

[GH13]'s second step is to apply a transformation that takes a list-decoding protocol to a unique decoding protocol, incurring a blowup in the alphabet size. Since we are working over a binary alphabet, we cannot afford to apply this same second transformation. Instead, we notice that our inefficient protocol has a property that we call *scaling*. Essentially, this means that the amount of confidence Alice and Bob have in their final transcript guesses is directly related to the amount of corruption the adversary put in. More specifically, if the adversary corrupted $\frac{1}{6}-\rho$ of the communication ($\rho > 0$), then Alice and Bob end up with the correct transcript and are $\propto \rho$ confident in its correctness; and if the adversary corrupted $\frac{1}{6}+\rho$ of the communication, then Alice and Bob may end up with incorrect transcripts but they are only $\propto \rho$ confident. We can understand this as saying that $\frac{1}{6}-\rho$ corruption results in a net good confidence of ρ (where ρ can be positive or negative: $\rho < 0$ means that there was ρ confidence in a bad transcript).

This allows us to consider the same boosting transformation that [GH13] did, with the following caveat: whenever a simulated subprotocol results in a complete transcript, that transcript obtains

a vote *proportional to the confidence the parties have in the simulated protocol's correctness*. Then, if the adversary corrupts $< \frac{1}{6}$ of the protocol, the net good votes (i.e. the number of votes for the correct transcript minus the total number for all incorrect transcripts) must be positive, so Alice and Bob can determine the correct transcript.

4.5 Roadmap

In the next three chapters, we flesh out the construction overviewed in this chapter. In particular, Chapter 5 will show how to amplify the computational efficiency to quasi-linear from arbitrarily bad runtimes. Then, in Chapter 6, we introduce our concept of layered codes and prove the properties necessary for our protocol. Finally, Chapter 7 will put all these pieces together into our final efficient, optimally error resilient scheme.

Chapter 5

Boosting: Achieving Computational Efficiency

In this chapter, we show how to boost the computational efficiency of a scheme. Our boosted protocol draws inspiration from the list-decoding boosting scheme of [GH13], which drew ideas from [BK12]. We begin by recalling the necessary setup from [GH13].

5.1 The Simulation Paradigm of [GH13, BK12]

Assume that π_0 is an alternating binary protocol of length n_0 (any binary protocol can be made alternating by increasing the communication by at most a factor of 2). We can view π_0 as a *protocol tree* $\mathbb{T}$, in which the edges at odd levels correspond to Alice's messages and the edges at even levels correspond to Bob's messages. For any input x, π_0 defines a subset S_A of edges at the odd levels corresponding to Alice's possible responses, and similarly, for any input y, π_0 defines a subset S_B of edges at the even levels corresponding to Bob's possible messages. Note that for any (x, y), $S_A \cup S_B$ defines a unique rooted path $\mathcal{T}$ corresponding to the noiseless protocol $\pi_0(x, y)$. The goal is for both Alice and Bob to determine $\mathcal{T}$.

To do this, Alice and Bob each keep track of a set of edges $\mathcal{E}_A$ and $\mathcal{E}_B$. Initially both sets are empty. In each of many iterations, Alice (resp. Bob) will add some edges to $\mathcal{E}_A$ (resp. $\mathcal{E}_B$) extending some existing path in $\mathcal{E}_A$ (resp. $\mathcal{E}_B$). We remark that any new edges Alice adds must be consistent with her own behavior on her input x, i.e. she never adds an edge in an odd layer that does not belong to S_A. The same holds for Bob. It thus holds that at any point the unique longest rooted path in both $\mathcal{E}_A$ and $\mathcal{E}_B$ is a prefix of $\mathcal{T}$.

The process by which Alice and Bob add edges to their respective set in each iteration is as follows. They first run a subprotocol to determine their longest common rooted path. Then, they run the next $\log^4 n_0$ rounds of the noiseless protocol. They perform both these steps under a single error-resilient simulation. The idea is that every time not too many errors have happened in an iteration, both Alice and Bob add $\log^4 n_0$ edges to the correct path corresponding to $\mathcal{T}$.

If the longest common rooted path is a path from the root to a leaf, then Alice and Bob instead add some weight to that leaf. Over the course of many iterations, the hope is that the leaf with the largest weight at the end of the protocol should correspond to $\mathcal{T}$. We remark that [GH13] showed a list-guarantee assuming not too many errors occurred: at the end of this procedure, Alice and Bob

will each have a small list of leaves each containing the true leaf corresponding to $\mathcal{T}$. (They then need to run this procedure many times in parallel with sending an error correcting code in order for both parties to narrow down the correct transcript, resulting in an alphabet blowup.) For us, we will show that if our inefficient simulation has a property known as *scaling* (see Definition 5.2), then at the end of this procedure Alice and Bob will each have narrowed down to a *unique* leaf, precisely, the leaf corresponding to $\mathcal{T}$, provided not too many errors occurred.

The Tree-Intersection Problem. The problem of finding their longest shared path is called the *tree-intersection problem*. Precisely, assuming Alice and Bob have sets of edges $\mathcal{E}_A$ and $\mathcal{E}_B$ respectively each forming a rooted tree under the promise that $\mathcal{E}_A \cap \mathcal{E}_B$ is a rooted path, the problem is for Alice and Bob to recover this rooted path using as little communication and computation as possible.

In [GH13], they give a data structure for $\mathcal{E}_A$ and $\mathcal{E}_B$ that optimizes the computational complexity of a protocol solving the tree-intersection problem.

Theorem 5.1. *[GH13] There is an incremental data structure that maintains a rooted subtree of the rooted infinite binary tree under edge additions with amortized computational complexity of $\tilde{O}(1)$ time per edge addition. Furthermore, for any $c = \Omega(1)$ and given two trees of maximum size n maintained by such a data structure, there is a tree-intersection protocol that uses $100c\log^4 n$ rounds of communication over a noiseless binary channel, $O(c\log^4 n)$ bits of randomness, and $\tilde{O}(1)$ computation steps to solve the tree intersection problem, that is, find the intersection path with failure probability at most $2^{-c\log^4 n}$.*

5.2 Scaling Schemes

We now define precisely what we mean by a *scaling* scheme. Intuitively, a scaling scheme is a scheme in which Alice and Bob output a *confidence* in addition to a transcript. This confidence should give a bound on the total error in the protocol. For instance, if there is no corruption, then Alice and Bob should output the correct transcript with large confidence. If there is some corruption, then Alice and Bob should output the correct transcript with smaller confidence. If there is too much corruption, then Alice and Bob may output an incorrect transcript, but their confidence cannot exceed a certain quantity specified by the amount of error that occurred (i.e. if the adversary wishes Alice and Bob to be more confident in an incorrect transcript, she must corrupt more of the protocol).

Definition 5.2 ($(\rho, \epsilon, \mu_\epsilon)$-Scaling Schemes)**.** *A scheme for simulating a noiseless protocol of length n is* $(\rho, \epsilon, \mu_\epsilon)$-scaling *if, at the end of the protocol, Alice and Bob output guesses T_A and T_B for the noiseless transcript $\mathcal{T}$ along with confidences $c_A, c_B \in [0, 1]$, with the following guarantees:*

- ***f** Consistency: All of Alice's messages in T_A are consistent with her behavior in π_0 on input x. Similarly, all of Bob's messages in T_B are consistent with his behavior in π_0 on input y.*
- ***f** Scaling 1: If a $\delta < (1-\epsilon)\cdot\rho$ fraction of the scheme was corrupted, then*

$$\Pr\left[T_A = T_B = \mathcal{T} \ \wedge \ c_A, c_B \geq 1 - \frac{\delta}{\rho} - \epsilon\right] \geq 1 - \mu_\epsilon(n).$$

- **f** *Scaling 2: If $\delta \geq (1-\epsilon)\cdot\rho$ fraction of the scheme was corrupted, then*

$$\Pr\left[\left(T_A \neq \mathcal{T} \ \wedge \ c_A > \frac{\delta}{\rho} - 1 + \epsilon\right) \vee \left(T_B \neq \mathcal{T} \ \wedge \ c_B > \frac{\delta}{\rho} - 1 + \epsilon\right)\right] \leq \mu_\epsilon(n).$$

5.3 Boosting

f Protocol 5-1 : Boosting

Let $\mathcal{P}'$ be a $(\rho, \epsilon, \mu_\epsilon)$-scaling scheme that simulates noiseless protocols of length n' by a protocol of length $r_\epsilon(n')$ that has computational complexity $T_\epsilon(n')$. Choose $C_\epsilon \geq 100/\epsilon + 1$.
For a protocol π_0 that has length n_0, and on inputs (x, y), Alice and Bob run the following scheme:

1. Alice and Bob each keep track of a list $\mathcal{E}_A, \mathcal{E}_B \subseteq \mathbb{T}$ of edges they have simulated so far, using the data structure from 5.1. Initially, $\mathcal{E}_A, \mathcal{E}_B = \emptyset$. They also each keep track of a dictionary[a] $\mathcal{L}_A, \mathcal{L}_B$ of leaves, i.e. full transcripts T of $\mathbb{T}$, mapping to $\mathbb{R}_{\geq 0}$. Initially, for any full transcript T of $\mathbb{T}$, $\mathcal{L}_A[T] = \mathcal{L}_B[T] = 0$.
2. For $i = 1, \ldots, \frac{n_0}{\epsilon \log^4 n_0} =: \beta$, they use $\mathcal{P}'$ to simulate the following $n' = C_\epsilon \cdot \log^4 n_0$ round noiseless protocol:
   (a) Alice and Bob run the tree-intersection protocol given in Theorem 5.1, using $(C_\epsilon - 1)\log^4 n_0$ rounds and $\tilde{O}(1)$ computation steps. At the end, with probability $1 - 2^{-((C_\epsilon-1)/100)\cdot\log^4 n_0} \geq 1 - 2^{-\log^4 n_0/\epsilon}$, the two parties have determined the common rooted path $p = \mathcal{E}_A \cap \mathcal{E}_B$.
   (b) After Alice and Bob have determined a common path p, they fix p to be the transcript prefix of π_0 so far and run the next $\log^4 n_0$ rounds of π_0. (If there are fewer than $\log^4 n_0$ rounds in π_0 remaining after p, they treat the remaining rounds as sending all 0's.)

   At the end of the simulation, Alice has determined a transcript prefix $p_A \subseteq \mathcal{E}_A$ along with up to $\log^4 n_0$ subsequent edges extending p_A. She also has a confidence $c_A \in [0, 1]$. She adds the $\leq \log^4 n_0$ edges to $\mathcal{E}_A$ (ignoring duplicates). Further, if p_A is a complete transcript of length n_0, she adds c_A to $\mathcal{L}_A[p_A]$. Bob does the same.
3. At the end of the protocol, let $T_A = \arg\max_p \mathcal{L}_A[p]$ be the transcript with the highest weight in $\mathcal{L}_A$, and let $w_A = \mathcal{L}_A[T_A]$. Also, let $w_A^c = \sum_{p \neq T_A} \mathcal{L}_A[p]$ be the total weight assigned to all the other leaves excluding T_A. Then, Alice outputs T_A, along with confidence $c_A = \frac{w_A - w_A^c}{\beta}$.

   Similarly, Bob outputs the transcript $T_B = \arg\max_p \mathcal{L}_B[p]$ and confidence $c_B = \frac{w_B - w_B^c}{\beta}$, where $w_B = \mathcal{L}_B[T_B]$ and $w_B^c = \sum_{p \neq T_B} \mathcal{L}_B[p]$ is the total weight on all the other leaves excluding T_B.

[a] Roughly, a dictionary is implemented by a hash table.

Theorem 5.3. *Let $\epsilon < 0.25$ and $C_\epsilon \geq 100/\epsilon + 1$. Assume a $(\rho, \epsilon, \mu_\epsilon)$-scaling scheme that simulates noiseless protocols of length n with communication complexity $r_\epsilon(n)$ and computational complexity $T_\epsilon(n)$. Then, the protocol given in Protocol 5-1 is a $(\rho, 4\epsilon, e^{-\epsilon n_0/10\log^4 n_0})$-scaling scheme for noiseless protocols of length n_0 that has communication complexity $\frac{n_0}{\epsilon\log^4 n_0} \cdot r_\epsilon(C_\epsilon \cdot \log^4 n_0)$ and computational complexity $\tilde{O}_\epsilon(n_0) \cdot T_\epsilon(C_\epsilon \log^4 n_0)$, assuming that $\mu_\epsilon(C_\epsilon \log^4 n_0) < \frac{\epsilon}{4}$.*

Proof. Clearly, the communication complexity in Protocol 5-1 is $\frac{n_0}{\epsilon\log^4 n_0} \cdot r_\epsilon(C_\epsilon \log^4 n_0)$. As for the computational complexity, note that in each iteration, Alice needs to do $T_\epsilon(C_\epsilon \log^4 n_0)$ computations to obtain a transcript T' and a confidence c'. She may further have to update $\mathcal{L}_A[T]$ with the

confidence c', for some complete transcript T, which can be done in amortized $O(\log L)$ time since a dictionary is roughly implemented by a hash table, where L is an upper bound on the size of $\mathcal{L}_A$. Finally, at the end of the protocol, she can determine T_A, w_A, w_A^c by making a linear pass through $\mathcal{L}_A$. Thus, the total computational complexity is $\beta \cdot (T_\epsilon(C_\epsilon \log^4 n_0) + O(\log L)) + \tilde{O}(L)$. Since $L \leq \beta$, which follows from the fact that Alice makes at most one value of $\mathcal{L}_A[p]$ nonzero in each iteration, the total computational complexity is $\tilde{O}(\beta) \cdot T_\epsilon(C_\epsilon \log^4 n_0)$ which is at most $\tilde{O}_\epsilon(n_0) \cdot T_\epsilon(C_\epsilon \log^4 n_0)$.

We will now show that our scheme is $(\rho, 4\epsilon, e^{-\epsilon n_0/10 \log^4 n_0})$-scaling. First, the consistency property follows because each of the protocols in the β iterations are consistent: Alice and Bob only add edges to $\mathcal{E}_A, \mathcal{E}_B$ that are consistent with their own input, so only transcripts consistent with their own input can gain weight in $\mathcal{L}_A, \mathcal{L}_B$. The rest of this proof will show the scaling properties.

Let $\delta_1, \ldots, \delta_\beta$ be the fractional amount of corruption in each of the β simulations, so that the total fractional amount of error is $\delta = \frac{1}{\beta} \sum_{i=1}^{\beta} \delta_i$. Let $T'_{A,1}, \ldots, T'_{A,\beta}$ and $c'_{A,1}, \ldots, c'_{A,\beta}$ (resp. $T'_{B,1}, \ldots, T'_{B,\beta}$ and $c'_{B,1}, \ldots, c'_{B,\beta}$) be the transcripts and confidences Alice (resp. Bob) has at the end of each of the β simulations.

Denote by $E_i(T'_i)$ denote the event that in the transcript T'_i, Alice and Bob correctly determine their longest shared path $\mathcal{E}_A \cap \mathcal{E}_B$ and extend it by $\log^4 n_0$ bits (or send 0's once the total transcript exceeds length n_0).

Lemma 5.4. *The following holds for the simulation in the i'th iteration:*

- *If there are at most $\delta_i < (1-\epsilon) \cdot \rho$ errors, then*

$$\Pr\left[E_i(T'_{A,i}) \ \wedge \ E_i(T'_{B,i}) \ \wedge \ c'_{A,i}, c'_{B,i} \geq 1 - \frac{\delta_i}{\rho} - \epsilon\right] \geq 1 - \mu_\epsilon(C_\epsilon \cdot \log^4 n_0) - 2^{-c \log^4 n_0}.$$

- *If there are at least $\delta_i \geq (1-\epsilon) \cdot \rho$ errors, then*

$$\Pr\left[\left(\neg E_i(T'_{A,i}) \ \wedge \ c'_A > \frac{\delta_i}{\rho} - 1 + \epsilon\right) \vee \left(\neg E_i(T'_{B,i}) \ \wedge \ c'_B > \frac{\delta_i}{\rho} - 1 + \epsilon\right)\right] \leq \mu_\epsilon(C_\epsilon \cdot \log^4 n_0) + 2^{-c \log^4 n_0}.$$

Proof. First, suppose that $\delta_i < (1-\epsilon) \cdot \rho$. Let T_i^* denote the noiseless protocol in the i'th simulation. Note that with probability $e^{\log^4 n_0/\epsilon}$, T_i^* may not correctly determine Alice and Bob's longest shared path. In particular,

$$\begin{aligned}
&\Pr\ \left[\neg\left(E_i(T'_{A,i}) \ \wedge \ E_i(T'_{B,i}) \ \wedge \ c'_{A,i}, c'_{B,i} \geq 1 - \frac{\delta_i}{\rho} - \epsilon\right)\right] \\
&\leq \Pr\left[\neg E_i(T_i^*)\right] + \Pr\left[\neg\left(T'_{A,i} = T'_{B,i} = T_i^* \ \wedge \ c'_{A,i}, c'_{B,i} \geq 1 - \frac{\delta_i}{\rho} - \epsilon\right)\right] \\
&\leq 2^{\log^4 n_0/\epsilon} + \mu_\epsilon(C_\epsilon \cdot \log^4 n_0)
\end{aligned}$$

by Theorem 5.1 and Definition 5.2.

On the other hand, if $\delta_i \geq (1-\epsilon)\cdot\rho$, it holds that

$$\begin{aligned}
&\Pr\left[\left(\neg E_i(T'_{A,i}) \ \wedge \ c'_A > \frac{\delta_i}{\rho} - 1 + \epsilon\right) \vee \left(\neg E_i(T'_{B,i}) \ \wedge \ c'_B > \frac{\delta_i}{\rho} - 1 + \epsilon\right)\right] \\
&\leq \Pr[\neg E_i(T_i^*)] + \Pr\left[\left(T'_{A,i} \neq T_i^* \ \wedge \ c'_A > \frac{\delta_i}{\rho} - 1 + \epsilon\right) \vee \left(T'_{B,i} \neq T_i^* \ \wedge \ c'_B > \frac{\delta_i}{\rho} - 1 + \epsilon\right)\right] \\
&\leq 2^{\log^4 n_0/\epsilon} + \mu_\epsilon(C_\epsilon \cdot \log^4 n_0),
\end{aligned}$$

where the second line follows from considering the cases where $\neg E_i(T_i^*)$ and $E_i(T_i^*)$, and the third line follows from Theorem 5.1 and Definition 5.2. □

Let $I \subseteq [\beta]$ denote the iterations in which $< (1-\epsilon)\cdot\rho$ of the scheme was corrupted.

Lemma 5.5. *With probability $1 - e^{-\epsilon^2\beta/10}$, for all except at most $\epsilon\cdot\beta$ values of $i \in [\beta]$, it holds that either:*

(1) $i \in I$ and $E_i(T'_{A,i}) \ \wedge \ E_i(T'_{B,i}) \ \wedge \ c'_{A,i}, c'_{B,i} \geq 1 - \frac{\delta_i}{\rho} - \epsilon$,

(2) $i \in [\beta]\backslash I$ and $\left(E_i(T'_{A,i}) \vee c'_A \leq \frac{\delta_i}{\rho} - 1 + \epsilon\right) \wedge \left(E_i(T'_{B,i}) \vee c'_B \leq \frac{\delta_i}{\rho} - 1 + \epsilon\right)$.

Proof. By Lemma 5.4, one of the two conditions holds for every $i \in [\beta]$ with probability at least $1 - 2^{-\log^4 n_0/\epsilon} - \mu_\epsilon(C_\epsilon \cdot \log^4 n_0)$. This means that the expected number of i satisfying one of the two conditions is $\varpi \geq (1 - 2^{-\log^4 n_0/\epsilon} - \mu_\epsilon(C_\epsilon \log^4 n_0)) \cdot \beta$.

Let X denote the number of $i \in [\beta]$ satisfying one of the two conditions. By Chernoff,

$$\Pr[X < (1-\epsilon)\cdot\beta] \leq \Pr[X < (1-\epsilon/2)\cdot\varpi] \leq e^{-\epsilon^2\varpi/8} \leq e^{-\epsilon^2\beta/10},$$

where the first and last inequalities follow from the fact that $2^{-\log^4 n_0/\epsilon} + \mu_\epsilon(C_\epsilon \log^4 n_0) \leq 2^{-1/\epsilon} + \mu_\epsilon(C_\epsilon \log^4 n_0) < \frac{\epsilon}{4} + \frac{\epsilon}{4} = \frac{\epsilon}{2}$, so $(1-\epsilon/2)\cdot\beta < \varpi$. In particular, the first inequality follows from $(1-\epsilon)\beta < (1-\epsilon/2)^2\beta < (1-\epsilon/2)\varpi$, and the last inequality follows from $0.8\beta < (1-\epsilon/2)\beta < \varpi$. □

Let $\Gamma \subseteq I$ be the set of all i satisfying (1), and let $\Lambda \subseteq [\beta]\backslash I$ be the set of all i satisfying (2). Note that after the first $\frac{n_0}{\log^4 n_0}$ iterations in Γ, Alice and Bob are both guaranteed to have all edges in the correct transcript $\mathcal{T}$ in their edge lists $\mathcal{E}_A$ and $\mathcal{E}_B$. After that point, in every iteration in Γ, Alice and Bob both determine the correct transcript $\mathcal{T} = \mathcal{E}_A \cap \mathcal{E}_B$ and add $c'_{A,i}$ (resp. $c'_{B,i}$) to $\mathcal{L}_A[\mathcal{T}]$ (resp. $\mathcal{L}_B[\mathcal{T}]$). This means that at the end of the protocol,

$$\mathcal{L}_A[\mathcal{T}] \geq \sum_{i\in\Gamma} c'_{A,i} - \frac{n_0}{\log^4 n_0} \geq (1-\epsilon)\cdot|\Gamma| - \frac{1}{\rho}\cdot\sum_{i\in\Gamma}\delta_i - \frac{n_0}{\log^4 n_0},$$

and similarly

$$\mathcal{L}_B[\mathcal{T}] \geq (1-\epsilon)\cdot|\Gamma| - \frac{1}{\rho}\cdot\sum_{i\in\Gamma}\delta_i - \frac{n_0}{\log^4 n_0}.$$

Meanwhile, for each iteration in Λ, a weight of at most $c'_{A,i}$ (resp. $c'_{B,i}$) is added to a wrong leaf. Furthermore, a weight of at most 1 is added to a wrong leaf for each iteration in $[\beta]\backslash(\Gamma\cup\Lambda)$, which

by Lemma 5.5 has size at most $\epsilon\beta$ with probability $1-e^{-\epsilon^2\beta/10}$. Thus, with probability $1-e^{-\epsilon^2\beta/10}$, the total weight on all the wrong leaves in Alice's tree is at most

$$\leq \sum_{i\in\Lambda} c'_{A,i}\cdot \mathbb{1}[T'_{A,i}\neq T^*_i] + \sum_{i\in[\beta]\setminus(\Gamma\cup\Lambda)} 1 \leq \frac{1}{\rho}\cdot\sum_{i\in\Lambda}\delta_i - (1-\epsilon)\cdot|\Lambda| + \epsilon\beta,$$

and simultaneously the total weight on all the wrong leaves in Bob's tree is at most

$$\leq \sum_{i\in\Lambda} c'_{B,i}\cdot \mathbb{1}[T'_{B,i}\neq T^*_i] + \sum_{i\in[\beta]\setminus(\Gamma\cup\Lambda)} 1 \leq \frac{1}{\rho}\cdot\sum_{i\in\Lambda}\delta_i - (1-\epsilon)\cdot|\Lambda| + \epsilon\beta.$$

Then, with probability $1-e^{-\epsilon^2\beta/10}$, the difference between the weight on the correct leaf and the combined weight on all the wrong leaves, for both Alice and Bob, is

$$\begin{aligned}
&\mathcal{L}_A[\mathcal{T}] - \sum_{T\neq\mathcal{T}}\mathcal{L}_A[T] \text{ (resp. } \mathcal{L}_B[\mathcal{T}] - \sum_{T\neq\mathcal{T}}\mathcal{L}_B[T])\\
&\geq \left[(1-\epsilon)\cdot|\Gamma| - \frac{1}{\rho}\cdot\sum_{i\in\Gamma}\delta_i - \frac{n_0}{\log^4 n_0}\right] - \left[\frac{1}{\rho}\cdot\sum_{i\in\Lambda}\delta_i - (1-\epsilon)\cdot|\Lambda| + \epsilon\beta\right]\\
&= (1-\epsilon)\cdot(|\Gamma|+|\Lambda|) - \epsilon\beta - \frac{1}{\rho}\cdot\sum_{i\in\Gamma\cup\Lambda}\delta_i - \frac{n_0}{\log^4 n_0}\\
&\geq (1-\epsilon)\cdot(\beta-\epsilon\beta) - \epsilon\beta - \frac{\delta\beta}{\rho} - \epsilon\beta\\
&\geq \left(1-\frac{\delta}{\rho}-4\epsilon\right)\cdot\beta, \qquad (5.1)
\end{aligned}$$

where we used that $\beta = \frac{n_0}{\epsilon\log^4 n_0}$ and that $\sum_{i\in\Gamma\cup\Lambda}\delta_i \leq \sum_{i\in[\beta]}\delta_i = \delta\beta$.

In particular, if $\delta < (1-\frac{\delta}{\rho}-4\epsilon)\cdot\rho$, then with probability $1-e^{-\epsilon^2\beta/10}$, both Alice and Bob output $T_A = T_B = \mathcal{T}$ and confidence $c_A, c_B \geq 1-\frac{\delta}{\rho}-4\epsilon$.

On the other hand, Equation 5.1 tells us that with probability $1-e^{-\epsilon^2\beta/10}$, for both Alice and Bob, for *any* incorrect leaf T_0, the total weight on T_0 minus the combined weight on all the other leaves is at most

$$\leq \left(\frac{\delta}{\rho}-1+4\epsilon\right)\cdot\beta,$$

since $\mathcal{L}_A[T_0] \leq \sum_{T\neq\mathcal{T}}\mathcal{L}_A[T]$, and $\sum_{T\neq T_0}\mathcal{L}_A[T] \geq \mathcal{L}_A[\mathcal{T}]$ (and same for Bob). Thus, in the case that $\delta > (1-\frac{\delta}{\rho}-4\epsilon)\cdot\rho$ of the entire protocol is corrupted, it holds with probability $1-e^{-\epsilon^2\beta/10}$ that either $T_A = \mathcal{T}$, or $T_A \neq \mathcal{T}$ and $c_A \leq \frac{\delta}{\rho}-1+4\epsilon$, and same for Bob.

It follows that Protocol 5-1 is $(\rho, 4\epsilon, e^{-\epsilon^2\beta/10}) = (\rho, 4\epsilon, e^{-\epsilon n_0/10\log^4 n_0})$-scaling.

□

Chapter 6

Layered Codes

In this section, we introduce *sensitive layered codes*, which are a generalization and strengthening of list tree codes to codes on layered graphs. List tree codes were first introduced in [BE14] as an analogue of list-decodable error correcting codes for the tree code setting. Sensitive layered codes are instead defined on certain graphs, and have list size 1 for most locations.

We first define suffix distance.

Definition 6.1 (Suffix Distance)**.** *For two strings $x, y \in \Sigma^n$, we define the suffix distance as follows:*

$$\Delta_{sfx}(x, y) = \max_{0 \le i \le n-1} \frac{\Delta(x[i+1:n], y[i+1:n])}{n-i}.$$

6.1 Layered Codes

Definition 6.2 (Layered Graph Over An Alphabet)**.** *Let Σ be an alphabet. A* layered graph over Σ of depth n *is a directed graph G that satisfies the following properties:*

- *The vertices of G can be split up into layers $0, 1, \ldots, n$. There is exactly one vertex in layer 0.*
- *Each vertex in layer $i < n$ has out-degree exactly $|\Sigma|$: it has $|\Sigma|$ children in layer $i+1$, where the $|\Sigma|$ out-edges are associated with not necessarily distinct elements of Σ.*

If G is a layered graph over Σ_{in} of depth n, note that any path p in G from the root node to a vertex in layer i can be associated with a string $\in \Sigma_{in}^i$. Likewise, any string $\in \Sigma_{in}^i$ corresponds to a unique path in G from the root node to a vertex in layer i. We will interchangeably refer to the path p or the associated string $\in \Sigma_{in}^i$. Furthermore, for any string $p \in \Sigma_{in}^i$, we use $v(p)$ to denote the vertex at the end of p.

Definition 6.3 (Layered Code)**.** *Let G be a layered graph over Σ_{in} of depth n. A* layered code C *of G with the alphabet Σ_{out} is an assignment of elements of Σ_{out} to the edges of G. We refer to such an assignment as a (G, Σ_{out})-code.*

For any subgraph $H \subseteq G$, we define $\mathsf{C}(H)$ to be the subgraph H inheriting labels from C. Specifically, for a rooted path $p \in \Sigma_{in}^i$, $\mathsf{C}(p) \in \Sigma_{out}^i$ is the string of i labels of the edges in p.

6.2 Prefix Trees

For any (G, Σ_{out})-code, any ϵ, and any word $w \in \Sigma_{in}^n$, let the list $L_i(\mathsf{C}, w, \epsilon)$ be the list of nodes in layer i that are the endpoint of at least one path whose encoding under C is close to the prefix of w of length i in their suffix distance. That is,

$$L_i(\mathsf{C}, w, \epsilon) = \{v(p) : p \in \Sigma_{in}^i \text{ s.t. } \Delta_{sfx}(\mathsf{C}(p), w[1:i]) < 1 - \epsilon\}.$$

We also write $L(\mathsf{C}, w, \epsilon) = \cup_{i=1}^n L_i(\mathsf{C}, w, \epsilon)$.

Consider a subset $S \subseteq L(\mathsf{C}, w, \epsilon)$. For each $v \in S$, we pick a path p from the root to v satisfying $\Delta_{sfx}(\mathsf{C}(p), w[1:|p|]) < 1 - \epsilon$. If these paths form a rooted tree, we call their union a *prefix tree* of S. We denote by $\mathcal{PT}(\mathsf{C}, w, \epsilon)$ the set of all prefix trees of all subsets of $L(\mathsf{C}, w, \epsilon)$.

Lemma 6.4. *Fix $w \in \Sigma_{out}^n$ and $\epsilon > 0$. For any subset $S \subseteq L(\mathsf{C}, w, \epsilon)$, there is a prefix tree of S.*

Proof. For a path q of length k, we define the *deficit* of q, denoted $\mathsf{deficit}(q)$, to be $\max_{0 \le j < k} [\Delta(\mathsf{C}(q)[j+1:k], w[j+$ For a path p of length i, we say that the *excess* of p at $k \le i$ is $(1-\epsilon)\cdot(i-k) - \Delta(\mathsf{C}(p)[k+1:i], w[k+1:i])$, denoted $\mathsf{excess}_k(p)$. Note that for any path p for which $v(p) \in L_i(\mathsf{C}, w, \epsilon)$, it holds that $\mathsf{excess}_k(p) > 0$ for any $k \le i$.

Furthermore, we claim that for any $p \in \Sigma_{in}^i$ such that $v(p) \in L_i$, letting p' denote the path obtained by replacing the first k edges by $q \in \Sigma_{in}^k$, we have that $\Delta_{sfx}(\mathsf{C}(p'), w[1:i]) < 1 - \epsilon$ iff $\mathsf{deficit}(q) < \mathsf{excess}_k(p)$. To see this, we can write

$$\Delta_{sfx}(\mathsf{C}(p'), w[1:i]) = \max\left\{ \begin{array}{r} \Delta_{sfx}(\mathsf{C}(p)[k+1:i], w[k+1:i]), \\ \max\limits_{0 \le j < k} \dfrac{\Delta(\mathsf{C}(p)[k+1:i], w[k+1:i]) + \Delta(q[j+1:k], w[j+1:k]))}{i-j} \end{array} \right\}.$$

Note that $\Delta_{sfx}(\mathsf{C}(p)[k+1:i], w[k+1:i]) < 1 - \epsilon$ because $v(p) \in L_i$. Thus, $\Delta_{sfx}(\mathsf{C}(p'), w[1:i]) < 1 - \epsilon$ iff

$$\Delta(\mathsf{C}(p)[k+1:i], w[k+1:i]) + \Delta(q[j+1:k], w[j+1:k])) < (1-\epsilon)\cdot(i-j)$$

for all $0 \le j < k$, or equivalently,

$$\mathsf{deficit}(q) < \mathsf{excess}_k(p).$$

Now, given a selection of paths $\{p(v)\}_{v \in S}$, where $p(v)$ connects the root to v, for each $k \in [n]$ define $\Lambda_k(p)$ to be the set of vertices $y \in G$ in layer k such that there are two paths $p(v)$ and $p(v')$, where $v \ne v' \in S$, for which $v(p(v)[1:k]) = v(p(v')[1:k]) = y$ but $p(v)[1:k] \ne p(v')[1:k]$. We define $\Psi(p)$ to be $(k_{max}, |\Lambda_{k_{max}}(p)|)$, with the lexicographical ordering, where k_{max} is the largest layer k for which $\Lambda_k(p)$ is nonempty.

In order to construct a prefix tree of S, we begin by choosing a path $p(v)$ from the root to v for each $v \in S$. Next, we perform an operation to p that decreases $\Psi(p)$, while preserving that p satisfies $\Delta_{sfx}(\mathsf{C}(p(v)), w[1:|p(v)|]) < 1 - \epsilon$ for all $v \in S$. The operation we perform is as follows: Choose $y_{max} \in \Lambda_{k_{max}}(p)$. Furthermore, let $v_1, \ldots, v_m \in S$ be such that $v(p(v_\iota)[1:k]) = y_{max}$. Define $q_\iota := p(v_\iota)[1:k]$ for each $\iota \in [m]$. Let $\hat{\iota} = \arg\min_{\iota \in [m]} \mathsf{deficit}(q_\iota)$, and let $q = q_{\hat{\iota}}$. Then, for each $\iota \in [m]$, we replace $p(v_\iota)$ with the path $p'(v_\iota) = q || p(v_\iota)[k+1:|p(v_\iota)|]$. Since $\mathsf{deficit}(q) \le \mathsf{deficit}(q_\iota)$, it holds that $\Delta_{sfx}(\mathsf{C}(p'(v_\iota)), w[1:|p'(v_\iota)|]) < 1 - \epsilon$ for all $\iota \in [m]$. (For all other $v \in S$ where $p(v)$ doesn't pass through y_{max}, we define $p'(v) = p(v)$.)

Note that $\Lambda_k(p')$ where $k > k_{max}$ must still be empty, as we have only altered edges in layers at most k_{max}. Furthermore, $|\Lambda_{k_{max}}(p')|$ is strictly less than $|\Lambda_{k_{max}}(p)|$, since we have replaced paths going through y_{max} with paths going through y_{max} so no new intersections in layer k_{max} were created, and we have removed y_{max} from $\Lambda_{k_{max}}(p)$. Thus, $\Psi(p') < \Psi(p)$. Also note that as long as $\Psi(p) > (0,0)$, we can continue this operation, so eventually $\Psi(p) = (0,0)$, at which point the union of $p(v), v \in S$ is a tree. □

For a subgraph H of G of depth at most $|w|$, we denote by $w(H)$ the graph where we write $w[i]$ on all edges at depth i. For a (G, Σ_{out})-code C, recall that $\mathsf{C}(H)$ is the subgraph H inheriting labels from C. For two labelings w and C of a subgraph H, we define $agr(w(H), \mathsf{C}(H))$ to be the number of edges of H for which the labels are the same.

Lemma 6.5. *For any $w \in \Sigma_{out}^n$ and $\epsilon > 0$, and for any $PT \in \mathcal{PT}(\mathsf{C}, w, \epsilon)$,*

$$agr(\mathsf{C}(PT), w(PT)) > \epsilon |PT|.$$

Proof. First, note that by definition of $L(\mathsf{C}, w, \epsilon)$, for any path p ending at $v \in L(\mathsf{C}, w, \epsilon)$ and not necessarily starting at the root, it holds that $agr(\mathsf{C}(p), w(p)) > \epsilon |p|$. We call this Property A.

We prove the lemma by induction on the number of leaves. If PT has only 1 leaf, then it is a path from root to leaf, and by Property A, $agr(\mathsf{C}(PT), w(PT)) > \epsilon |PT|$. Now, if PT has more than one leaf, let p be a branch of PT (i.e. a path from a vertex v_0 to a leaf v, where v_0 has more than one child). Then $PT \backslash p$ has one fewer leaf than PT, and by inductive hypothesis we have

$$agr(\mathsf{C}(PT \backslash p), w(PT \backslash p) > \epsilon(|PT| - |p|).$$

Furthermore, by Property A, we have that $agr(\mathsf{C}(p), w(p)) > \epsilon |p|$. Therefore,

$$agr(\mathsf{C}(PT), w(PT)) = agr(\mathsf{C}(PT \backslash p), w(PT \backslash p) + agr(\mathsf{C}(p), w(p)) > \epsilon |PT|.$$

□

6.3 Sensitive Layered Codes

Definition 6.6 (Sensitive Layered Code)**.** *Let G be a layered graph over Σ_{in} of depth n. A* ϵ-sensitive layered code *for G and alphabet Σ_{out} is a (G, Σ_{out})-code such that for all $w \in \Sigma_{out}^n$ and all $PT \in \mathcal{PT}(\mathsf{C}, w, \epsilon)$,*

$$agr(\mathsf{C}(PT), w(PT)) \leq (1 + \epsilon)n. \tag{6.1}$$

Theorem 6.7. *For $\epsilon \in (0, \frac{1}{2})$ and a layered graph G over Σ_{in} with depth $n \geq \frac{2}{1-\epsilon}$, let $|\Sigma_{out}| > 2|\Sigma_{in}|)^{6/\epsilon^2}$. Then, a random (G, Σ_{out})-code is a ϵ-sensitive layered code on G with alphabet Σ_{out} with probability at least $1 - 2^{-n/4\epsilon}$.*

The proof of Theorem 6.7 essentially follows from the proof of Theorem 22 in [BE14]. To prove it, we will need the following two lemmas:

Lemma 6.8. *If G is a layered graph over Σ_{in}, there exist at most $(|\Sigma_{in}| + 1)^{2s}$ rooted subtrees of G of size s.*

Proof. Consider the path obtained by conducting a DFS on a rooted subtree, where each symbol indicates which child to go to, and $|\Sigma_{in}|+1$ indicates to go back up the edge traversed downwards to get to the current vertex (note that this edge is unique since we only traverse a subtree). Then, each edge in the subtree is traversed twice. Thus, the number of rooted subtrees of G is at most $(|\Sigma_{in}|+1)^{2s}$. □

Lemma 6.9. *For any $w \in \Sigma_{out}^n$ and for any collection PT of s edges of G, it holds that*

$$Pr[agr(\mathsf{C}(PT), w(PT)) \geq \epsilon s] \leq |\Sigma_{out}|^{-\epsilon s}\binom{s}{\epsilon s} \leq |\Sigma_{out}|^{-\epsilon s}2^s,$$

where randomness is taken over the random choice of layered code C on G with Σ_{out}).

Proof. The first inequality follows from the union bound over all possible locations where $\mathsf{C}(PT)$ and $w(PT)$ agree, and the second inequality follows from $\binom{s}{\epsilon s} \leq 2^s$. □

Proof of Theorem 6.7. If $w \in \Sigma_{out}^n$ violates (6.1), then there is a prefix tree PT of a subset $S \subseteq L(\mathsf{C}, w, \epsilon)$ such that $agr(\mathsf{C}(PT), w(PT)) > \max\{\epsilon|PT|, (1+\epsilon)n\}$, where $agr(\mathsf{C}(PT), w(PT)) > \epsilon|PT|$ is given by Lemma 6.5. To show that such w does not exist, we will show that with high probability over the choice of a random (G, Σ_{out})-code, $agr(\mathsf{C}(PT), w(PT)) \leq \max\{\epsilon|PT|, (1+\epsilon)n\}$ for all rooted subtrees PT and $w \in \Sigma_{out}^n$. It is enough to prove this claim for all $|PT| \geq (1+\frac{1}{\epsilon})n$, since if $|PT| < (1+\frac{1}{\epsilon})n$, then we can extend PT to a tree PT' of size $(1+\frac{1}{\epsilon})n$ and for this subtree it will hold that $agr(\mathsf{C}(PT'), w(PT')) \leq (1+\epsilon)n$ and thus $agr(\mathsf{C}(PT), w(PT)) \leq (1+\epsilon)n$. We thus seek to show that with high probability over the choice of a random layered code, $agr(\mathsf{C}(PT), w(PT)) \leq \epsilon|PT|$ for all rooted subtrees PT of size $\geq (1+\frac{1}{\epsilon})n$ and $w \in \Sigma_{out}^n$.

Using Lemmas 6.8 and 6.9, we union bound over all possible trees of size $\geq (1+\frac{1}{\epsilon})n =: s$ and words w to see that the probability there exists $|PT| \geq (1+\frac{1}{\epsilon})n$, $w \in \Sigma_{out}^n$ for which $agr(\mathsf{C}(PT), w(PT)) \geq \epsilon s$ is upper bounded by

$$\begin{aligned}\sum_{s=(1+\frac{1}{\epsilon})n}^{\infty} |\Sigma_{out}|^{-\epsilon s}2^s \cdot (|\Sigma_{in}|+1)^{2s} \cdot |\Sigma_{out}|^n &= |\Sigma_{out}|^n \sum_{s=(1+\frac{1}{\epsilon})n}^{\infty} \left(\frac{2\cdot(|\Sigma_{in}|+1)^2}{|\Sigma_{out}|^{\epsilon}}\right)^s \\ &\leq |\Sigma_{out}|^n \sum_{s=(1+\frac{1}{\epsilon})n}^{\infty} \left(\frac{8\cdot|\Sigma_{in}|^2}{|\Sigma_{out}|^{\epsilon}}\right)^s\end{aligned}$$

Since $|\Sigma_{out}| > (2|\Sigma_{in}|)^{6/\epsilon^2} > 8|\Sigma_{in}|^2$, this is upper bounded by

$$\begin{aligned}\leq |\Sigma_{out}|^n \left(\frac{8\cdot|\Sigma_{in}|^2}{|\Sigma_{out}|^{\epsilon}}\right)^{(1+\frac{1}{\epsilon})n-1} &= \frac{(8\cdot|\Sigma_{in}|^2)^{(1+\frac{1}{\epsilon})n-1}}{|\Sigma_{out}|^{\epsilon n-\epsilon}} \\ &\leq \frac{(8\cdot|\Sigma_{in}|^2)^{(1+\frac{1}{\epsilon})n-1}}{(2\cdot|\Sigma_{in}|)^{6(n-1)/\epsilon}} \\ &\leq \frac{(8\cdot|\Sigma_{in}|^2)^{(1+\frac{1}{\epsilon})n-1}}{(8\cdot|\Sigma_{in}|^2)^{2(n-1)/\epsilon}} \\ &\leq \left(8\cdot|\Sigma_{in}|^2\right)^{-((1-\epsilon)n-2)/\epsilon} \\ &\leq 2^{-n/4\epsilon},\end{aligned}$$

where in the last line we use that $\epsilon < \frac{1}{2}$ and $(1-\epsilon)n \geq 2$. □

6.4 Decoding

Sensitive (G, Σ_{out}) codes will be useful for us because they guarantee that for most locations i on which $\mathsf{C}(x)$ and w agree, $w[1:i]$ decodes to $v(x[1:i])$. First, we define decoding.

Definition 6.10 (CDec)**.** *Given an ϵ-sensitive-(G, Σ_{out})-code C, we define CDec to be the algorithm that takes as input a string $w \in \Sigma_{out}^i$ and outputs $v \in G$ such that there exists a path $p \in \Sigma_{in}^i$ satisfying $\Delta(\mathsf{C}(p), w) < 1-\epsilon$ if exactly one such v exists, and $\perp$ otherwise.*

The main theorem of this section is the following:

Theorem 6.11. *For every ϵ, n, for any layered graph over Σ_{in} of depth n and any ϵ-sensitive-(G, Σ_{out})-code $\mathsf{C} : \Sigma_{in}^n \to \Sigma_{out}^n$, and for any $x \in \Sigma_{in}^n$ and $w \in \Sigma_{out}^n$, let J be the set of indices where $\mathsf{C}(x)[i] = w[i]$. For all but at most $2\epsilon n$ values of $i \in J$, it holds that $\mathsf{CDec}(w[1:i]) = v(x[1:i])$.*

We defer the proof of Theorem 6.11 to after we state a few lemmas.

Lemma 6.12. *Given an ϵ-sensitive-(G, Σ_{out})-code C, for any $w \in \Sigma_{out}^n$ and $\epsilon > 0$, it holds that $|L_i(\mathsf{C}, w, \epsilon)| \leq 1$ for at least $(1-\epsilon)n$ values of $i \leq n$.*

Proof. Given w, we construct w' as follows. Pick a prefix tree PT of $L(\mathsf{C}, w, \epsilon)$. For every $i \leq n$, define $PT_i(w)$ to be the set of edges in the i'th layer of PT. If for all $e \in PT_i(w)$ we have that $\mathsf{C}(e) \neq w[i]$, then set $w'[i]$ to be $\mathsf{C}(e)$ for some arbitrary $e \in PT_i(w)$. Otherwise, set $w'[i] = w[i]$.

Notice that $L(\mathsf{C}, w, \epsilon) \subseteq L(\mathsf{C}, w', \epsilon)$, since the only indices of w that were changed were those that did not agree with any of the labels of PT in the corresponding layer, so for any path $p(v) \subseteq PT, v \in L_i(\mathsf{C}, w, \epsilon)$, it holds that $\Delta_{sfx}(\mathsf{C}(p(v)), w'[1:|p(v)|]) \leq \Delta_{sfx}(\mathsf{C}(p(v)), w[1:|p(v)|]) < 1-\epsilon$. This means that $PT \in \mathcal{PT}(\mathsf{C}, w', \epsilon)$. But by the definition of an ϵ-sensitive-(G, Σ_{out})-code (Definition 6.6),

$$agr(\mathsf{C}(PT), w'(PT)) \leq (1+\epsilon)n.$$

On the other hand, we constructed w' so that in each layer i, there is at least one edge on which C and w' agree. Therefore, the number of layers in which there is more than 1 edge on which C and w' agree is $\leq \epsilon n$. In other words, the number of layers in which there is at most 1 edge on which C and w' agree is at least $(1-\epsilon)n$. Let this set of layers be $I \subseteq [n]$.

Finally, note that for any vertex $v \in L_i(\mathsf{C}, w, \epsilon)$ and associated path $p(v) \subseteq PT$, it must hold that $\mathsf{C}(p(v))[i] = w[i] = w'[i]$ (otherwise the suffix distance of $\mathsf{C}(p(v))$ to w is 1), so for each of the $\geq (1-\epsilon)n$ layers in I, there is at most 1 vertex $v \in L_i(\mathsf{C}, w, \epsilon)$. □

Lemma 6.13 ([Gel17])**.** *For any $r, s \in \Sigma^n$, if $\Delta(r, s) = \beta n$, then there exists a set of indices $I \subseteq [n]$ of size $|I| \geq (1-\beta/\alpha)n$ such that for any $i \in I$,*

$$\Delta_{sfx}(r[1:i], s[1:i]) < \alpha.$$

Proof of Theorem 6.11. By Lemma 6.13, there exists a set of indices $I \subseteq [n]$ of size $|I| \geq (1 - \frac{1-|J|/n}{1-\epsilon})n = \frac{|J|-\epsilon n}{1-\epsilon} \geq |J| - \epsilon n$ such that for any $i \in I$, $\Delta_{sfx}(\mathsf{C}(x)[1:i], w[1:i]) < 1-\epsilon$. Note also that $I \subseteq J$, since if $\mathsf{C}(x)[i] \neq w[i]$, then $\Delta_{sfx}(\mathsf{C}(x)[1:i], w[1:i]) = 1$.

Furthermore, by Lemma 6.12, it holds that $|L_i(\mathsf{C}, w, \epsilon)| > 1$ on at most ϵn values. Thus, there are at least $|J| - 2\epsilon n$ values of J for which $\mathsf{CDec}(w[1:i]) = v(x[1:i])$. □

Remark 6.14. *In this section, we defined sensitive layered codes on finite-depth layered graphs. However, our proofs extend straightforwardly to give sensitive layered codes on layered graphs of* infinite depth*. For an infinite graph, sensitivity means that the restriction of the code to any depth n (above a certain threshold) should be a sensitive layered code. It is straightforward via a union bound to see that a random layered code on an infinite layered graph will, with positive probability, satisfy sensitivity.*

Chapter 7

Putting It All Together: Efficient and Optimally Error Resilient Interactive Communication for Bit Flip Errors

In this section, we will formally describe our algorithm to convert any noiseless interactive protocol between Alice and Bob to one that is resilient to $\frac{1}{6} - \epsilon$ bit flips for any sufficiently small $\epsilon > 0$ (say, $\epsilon < 0.01$), with constant multiplicative blowup in communication complexity and $\tilde{O}(|\pi_0|)$ computational complexity. We note that an error resilience of $\frac{1}{6}$ is known to be optimal (see Theorem **??**). We focus mainly on describing a computationally inefficient scheme, but a recursive application of Corollary 5.3 results in a computationally efficient scheme.

Throughout this section, let be π_0 the noiseless protocol of length n_0 that Alice and Bob are trying to simulate. Alice's and Bob's private inputs respectively are $x, y \in \{0,1\}^{n_{in}}$ for some $n_{in} \in \mathbb{N}$. We assume that π_0 is alternating (meaning that Alice speaks in the odd rounds and Bob speaks in the even: any protocol can be made alternating with at most a factor of 2 blowup in communication). We also assume that Alice's first message is a 1. The correct noiseless transcript for π_0 is denoted $\mathcal{T} = \mathcal{T}(x, y)$. We also define $f_x : \{0,1\}^s \to \{0,1\}$ to be the function taking a partial transcript with Bob as the last speaker (only defined on even s) and outputs Alice's next message if she has input x, as defined by the protocol π_0. Similarly, we define $f_y : \{0,1\}^s \to \{0,1\}$ to be the function taking a partial transcript with Alice as the last speaker and outputs Bob's next message on input y as defined by π_0. We say a transcript T is *inconsistent with* x if for some even s with $|s| < |T|$, if $f_x(T[1:s]) \neq T[s+1]$, and similarly *inconsistent with* y if for some odd s, $f_y(T[1:s]) \neq T[s+1]$.

We denote a parameter $\epsilon > 0$, where the adversary will be permitted to flip $\frac{1}{6} - O(\epsilon)$ bits.

7.1 Preliminaries and Definitions

In our protocol, Alice and Bob will each track a guess for the noiseless transcript $\mathcal{T}$. Specifically, they will track a sequence of updates denoted $U_A, U_B \in \{0, 1, \leftarrow, \bullet\}^*$ that evaluates to their current guess for $\mathcal{T}$. Generally, Alice's guess is odd length (meaning $|t(v(U_A))|$ is odd) since she speaks on odd turns in π_0, and Bob's guess $t(v(U_B))$ is even length. The exception is if Alice has a transcript that is either length 0 or length n_0. Roughly, an update of 0 or 1 adds this bit onto the transcript, an update of $\leftarrow$ rewinds the previous bit of the transcript, and an update of $\bullet$ keeps the transcript

the same. After each message, the receiving party will append some new updates to this sequence based on the other person's message. We begin with some necessary definitions.

7.1.1 Transcript Graph

We begin by informally describing the layered graph that the parties use to build their transcript guesses. The vertices of G at a given layer ℓ describe the possible transcript guesses for the noiseless protocol that a party could have after appending ℓ edges $\in \{0, 1, \leftarrow, \bullet\}^*$ as updates to the transcript guess. The depth of the graph is $K = \frac{n_0}{\epsilon}$.

Definition 7.1 (Transcript Graph (G))**.** *Let G be the following particular instance of a layered graph over the alphabet $\{0, 1, \leftarrow, \bullet\}$ (see Definition 6.2).*

- *At every layer $\ell \in [0, K]$, the vertices are all elements of the form $\{0,1\}^{\leq \ell}_{\ell}$ (for example, at layer 5, a possible vertex is 01_5). For a vertex v denoted $v = y_\ell$, where $y \in \{0,1\}^*$ and $\ell \in \mathbb{N}$, define $t(v) := y \in \{0,1\}^*$ and $\ell(v) := \ell$. The set of all vertices of G is denoted Π.*
- *The out-edges from a given node v in some layer $< K$ are $0, 1, \leftarrow, \bullet$. For an edge $e \in \{0, 1, \leftarrow, \bullet\}$, the node $v \oplus e$ at the end of the out-edge from v labeled e is computed as follows*

$$v \oplus e := \begin{cases} (t(v)||e)_{\ell(v)+1} & e \in \{0,1\} \\ (t(v)[1 : |t(v)| - 1])_{\ell(v)+1} & e = \leftarrow \text{ and } y \neq \emptyset \\ \emptyset_{\ell(v)+1} & e = \leftarrow \text{ and } t(v) = \emptyset \\ t(v)_{\ell(v)+1} & e = \bullet \end{cases}.$$

Vertices in layer K have no out-edges.

As shorthand, for a layered code C on G, and for $v \in \Pi$ and $p \in \Sigma^*$, let $\mathsf{C}(v, p) \in \Sigma^{|p|} := \mathsf{C}(H)$ where H is the subgraph of G corresponding to the path starting at v obtained by following the edges specified by p.

7.1.2 Transcript Operations and Instructions

Along with U_A and U_B, Alice and Bob track a weight (confidence) w_A and w_B associated with this guess. We will have that $w = 0$ unless T is a complete transcript. A message received from the other party will contain an *instruction* for how to update (U, w). The instruction is in $\{0, 1, \leftarrow, \bullet\}$.

We define some functions that describe the updates that Alice and Bob make to (U_A, w_A) and (U_B, w_B). We begin with the definition of $\mathsf{op}_x(T)$ and $\mathsf{op}_y(T)$. This function takes a partial transcript $T \in \{0,1\}^*$[1] and calculates the instruction that the party with x or y gives to extend T. The function is defined on every possible partial transcript T, but only takes on a meaningful value when the party with the corresponding x or y is the next to speak, or if the transcript is complete (of length n_0).

Definition 7.2 ($\mathsf{op}_r(T)$)**.** *We define $\mathsf{op}_r(T) : \{0,1\}^{\leq n_0} \to \{0, 1, \leftarrow\}$, for $r \in \{x, y\}$. Let the set S denote the set of lengths of T on which f_r is defined: S is all the even indices $< n_0$ if $r = x$ or all the odd indices $< n_0$ if $r = y$.*

[1]Notice that $T \in \{0,1\}^*$ while each party tracks $U \in \{0, 1, \leftarrow, \bullet\}^*$. Each U evaluates to a transcript $t(v(U)) \in \{0,1\}^*$ which corresponds to the input to op.

- *If T is inconsistent with r, then* $\mathsf{op}_r(T) = \leftarrow$.
- *Else if* $|T| \in S$, *then* $\mathsf{op}_r(T) = f_r(T)$.
- *Else,* $\mathsf{op}_r(T) = 1$.

The final condition which results in a "default" response of $\mathsf{op}_r(T) = 1$ occurs in one of two cases: when the party with input r is not the next to speak, allowing 1 to serve as a meaningless instruction, or when the transcript is complete (of length n_0) and the party wants to indicate it is consistent with their input.

Next, we define the function $\mathsf{op}_{T'}(T)$, where T' is a complete transcript. The function $\mathsf{op}_{T'}(T)$ takes a partial transcript T and returns the instruction that brings it one step closer to T'.

Definition 7.3 ($\mathsf{op}_{T'}(T)$)**.** *Let* $T' \in \{0,1\}^{\leq n_0}$ *with* $|T'| = n_0$. *We define* $\mathsf{op}_{T'}(T) : \{0,1\}^{\leq n_0} \to \{0, 1, \leftarrow\}$ *as follows.*

- *If* $T' = T$, *then* $\mathsf{op}_{T'}(T) = 1$.
- *Else, if* T *is a strict prefix of* T', *then* $\mathsf{op}_{T'}(T) = T'[|T| + 1]$.
- *Else,* $\mathsf{op}_{T'}(T) = \leftarrow$.

Next, we define a function that Alice and Bob use to update their transcript guess U_A or U_B and weight w_A or w_B when they receive an instruction. Every time a party receives a message, the party adds two edges onto their guess U_A or U_B: namely the update $\hat{\delta} \in \{0, 1, \leftarrow, \bullet\}$ that they deduce from the other party's message, and their own response to that addition.[2] Again, recall that Alice's partial transcript guess $t(v(U_A))$ is of odd or exactly 0 or n_0 length, and Bob's guess $t(v(U_B))$ is of even length.

Definition 7.4 ($(U, w) \otimes_r \hat{\delta}$)**.** *Let* $r \in \{x, y\}$. *Given a sequence of updates* $U \in \{0, 1, \leftarrow, \bullet\}^*$, *an instruction* $\hat{\delta} \in \{0, 1, \leftarrow, \bullet\}$, *and weight* $w \in \mathbb{N}$, *return a new pair* $(U', w') \leftarrow (U, w) \otimes_r \hat{\delta}$ *as follows. As before, let the set* S *denote the set of lengths of* $T \in \{0,1\}^*$ *on which* f_r *is defined:* S *is all the even indices* $< n_0$ *if* $r = x$ *and all the odd indices* $< n_0$ *if* $r = y$.

- *If* $\hat{\delta} = \bullet$*:*

  Let $U' = U|| \bullet ||\bullet$ *and* $w' = w$.
- *If* $\hat{\delta} = \leftarrow$*:*

  If $w > 0$, *then let* $U' = U|| \bullet ||\bullet$ *and* $w' = w - 1$.

  Otherwise, if $|t(v(U))| - 1 \in S$, *then let* $U' = U|| \leftarrow || \leftarrow$ *and* $w' = w$. *Else,* $|t(v(U))| \in S$, *and let* $U' = U|| \leftarrow ||\bullet$ *and* $w' = w$.
- *If* $\hat{\delta} = 0$ *or* $\hat{\delta} = 1$*:*

  Let $T = t(v(U))$. *If* $|T| = n_0$, *then* $U' = U|| \bullet ||\bullet$ *and* $w' = w + 1$.

  Otherwise, if $|T| - 1 \in S$*: if* $|T| < n_0 - 1$, *then* $U' = U||\hat{\delta}||\mathsf{op}_r(t(v(U||\hat{\delta})))$, *and if* $|T| = n_0 - 1$, *then* $U' = U||\hat{\delta}||\bullet$. *Else if* $|T| \in S$, *then* $U' = U|| \bullet ||\mathsf{op}_r(T)$. *In any case,* $w' = 0$.

Notice that in every case, the path U' is an extension of U with two additional letters.

[2] They will also add two more edges, corresponding to $\bullet\bullet$, to account for parity issues, but we leave this discussion for later. We also do not yet discuss how they deduce $\hat{\delta}$ from the other party's message.

7.1.3 The Error Correcting Code

Finally, we define the error correcting code ECC that Alice and Bob use to encode the letters of the large alphabet layered code.

Lemma 7.5 ([GZ22])**.** *There exists an explicit error correcting code*

$$\mathsf{ECC}_{\Sigma,\epsilon} := \Sigma^2 \times \{0, 1, \leftarrow, ?\} \to \{0,1\}^{M(|\Sigma|,\epsilon)}$$

for some $M(|\Sigma|, \epsilon) = O_\epsilon(|\Sigma|)$ with the following properties:

- *For any $z_0 \neq z_1 \in \Sigma^2$ and $\delta_0, \delta_1 \in \{0, 1, \leftarrow, ?\}$,*

$$\Delta\big(\mathsf{ECC}_{\Sigma,\epsilon}(z_0, \delta_0), \mathsf{ECC}_{\Sigma,\epsilon}(z_1, \delta_1)\big) \geq \left(\frac{1}{2} - \epsilon\right) \cdot M(|\Sigma|, \epsilon), \tag{7.1}$$

- *For any $z \in \Sigma^2$ and $\delta_0 \neq \delta_1 \in \{0, 1, \leftarrow, ?\}$,*

$$\Delta\big(\mathsf{ECC}_{\Sigma,\epsilon}(z, \delta_0), \mathsf{ECC}_{\Sigma,\epsilon}(z, \delta_1)\big) \geq \frac{2}{3} M(|\Sigma|, \epsilon). \tag{7.2}$$

We remark that due to the distance conditions, for any fixed z' and any string $s \in \{0,1\}^{M(|\Sigma|,\epsilon)}$, at most one of the following holds:

- There exists $\delta \in \{0, 1, \leftarrow, ?\}$ such that $\Delta(s, \mathsf{ECC}_{\Sigma,\epsilon}(z', \delta)) < \frac{1}{3}$.
- There exists $z \in \Sigma^2, \delta \in \{0, 1, \leftarrow, ?\}$ such that $\Delta(s, \mathsf{ECC}_{\Sigma,\epsilon}(z, \delta)) < \frac{1}{6} - \epsilon$.

In particular, the three cases in Protocol 7-1 are disjoint.

7.2 The Inefficient, Positive Rate Protocol

We are now ready to state our (inefficient) positive rate protocol that is resilient to $\frac{1}{6} - \epsilon$ errors.

Recall that π_0 is an alternating protocol of length n_0, such that Alice speaks first and her first message is always a 1. Let C be a ϵ-sensitive-(G, Σ)-code for some alphabet Σ of size $O_\epsilon(1)$. Note that Alice and Bob can agree on an explicit choice of C, for example by both choosing the lexicographically first such code (it takes up to 2^{2^K}-time to find such a code). Also let $\mathsf{ECC} = \mathsf{ECC}_{\Sigma,\epsilon}$ be the error correcting code from Lemma 7.5.

Before we state our protocol formally in Section 7.2.1, we give an explanation of the protocol. While Section 4.1 and Section 4.2 give an explanation of the ideas in our protocol, this section explains how we implement them. In this explanation, we first focus on when Eve corrupts a message either entirely to another valid message, or not at all. We talk about the protocol from Alice's perspective (Bob is symmetric).

Recall that Alice tracks a guess for the sequence of updates $U_A \in \{0, 1, \leftarrow, \bullet\}^*$ along with a confidence weight $w_A \geq 0$. The sequence of updates in U_A describes Alice's guess for the transcript: her transcript guess $\in \{0,1\}^{\leq n_0}$ is simply the result of applying the updates to the empty string.

Every round, Alice sends one of two things: she either asks her own question (a message of the form $\mathsf{ECC}(z, ?)$, where z lets Bob deduce U_A which specifies her transcript guess), or she sends an answer

to Bob's question (a message of the form $\mathsf{ECC}(z, \delta \in \{0, 1, \leftarrow\})$ where z reflects the transcript she believes Bob has asked about). Likewise, Bob always sends a question $\mathsf{ECC}(z, ?)$ or an answer $\mathsf{ECC}(z, \delta \in \{0, 1, \leftarrow\})$. We will discuss later what z should look like.

Whenever Alice receives a message $\mathsf{ECC}(z_B, \delta \in \{0, 1, \leftarrow, ?\})$ from Bob, she updates w_A and U_A based on the received message and history. She then chooses to send either a question or an answer. Specifically:

- If Alice receives an answer $\mathsf{ECC}(z_B, \delta \in \{0, 1, \leftarrow\})$ where z_B matches her own transcript guess, she updates (U_A, w_A) accordingly by setting $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \delta$. This consists of (with probability 1) appending two symbols to U_A and possibly adjusting the weight w_A so that she has overall updated in the direction specified by δ. She then asks a question.
- If she instead receives a question $\mathsf{ECC}(z_B, ?)$, she uses z_B and the history of received messages to make a guess for the full sequence of updates U_B^* that Bob has made. $T_B^* = t(v(U_B^*))$ is then her understanding of Bob's current transcript guess.
  - If T_B^* is a partial transcript or is inconsistent with x, she updates $(U_A, w_A) \leftarrow (U_A, W_A) \otimes_x \bullet$ ("do nothing"). She then sends an answer $\mathsf{ECC}(z_A, \delta = \mathsf{op}_x(T_B^*) \in \{0, 1, \leftarrow\})$.
  - Else if T_B^* is a complete transcript (length n_0) that is also consistent with x, she updates U_A with probability 0.5 in the direction of T_B^*, i.e. by computing $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \mathsf{op}_{T_B^*}(t(v(U_A)))$. This consists of appending two symbols to U_A and possibly adjusting w_A. She then asks a question.

  In the special case that $t(v(U_B)) =: T_B = T_A := t(v(U_A))$, i.e. Bob's current transcript guess is the same as Alice's (because Alice and Bob's transcripts are usually different parity lengths, this can only happen if $T_B = T_A$ are both the same complete transcript or both the empty transcript), Alice asks a question. Bob will interpret her question $\mathsf{ECC}(z_A, ?)$ as both an answer of 1 (extending his complete transcript guess or empty transcript) *and* a question. That is, if Bob receives Alice's message correctly, he will both update (U_B, w_B) (with probability 1) via the operation $\hat{\delta} = 1$ *and* send his question. Note that in both the case $T_B = T_A = \mathcal{T}$ or $T_B = T_A = \emptyset$ the update $\hat{\delta} = 1$ causes a good update, since we assumed Alice's first message is always a 1.

We emphasize that every time Alice updates (after receiving a message from Bob), she appends *two* elements $\in \{0, 1, \leftarrow, \bullet\}$ to U_A, so that the resulting transcript guess $t(v(U_A))$ still ends on her speaking. (The exception is when $t(v(U_A))$ is a complete transcript of length n_0 or the empty transcript of length 0: then, Alice still appends two update instructions, but the resulting transcript may be of even (n_0 or 0) length.)

The token z. When Alice is asking a question $\mathsf{ECC}(z, ?)$, we need z to allow Bob to determine Alice's current transcript guess $T_A = t(v(U_A))$. Note that sending $z = U_A$ (or even $z = T_A$) is too long. Instead, Alice simply sends $z \in \Sigma^2$ to be her most recent updates to U_A, i.e. the last two operations she appended to U_A, encoded into a tree code. Then many of Alice's messages (the ones where she asked a question) are symbols of the tree code encoding of U_A, which will be sufficient for Bob to determine U_A.

In the case where Alice answers Bob's question, her message is of the form $\mathsf{ECC}(z, \delta \in \{0, 1, \leftarrow\})$, where z must, in some way, echo Bob's question so that Bob can tell that she is answering the right question. As before, she cannot send z as the entire belief of Bob's transcript guess $t(v_B)$ where

$v_b \in \Pi$ is a vertex of G, because this is too long. Instead, z will be $\in \Sigma^2$ and will be dependent on her current belief about Bob's current transcript guess (as a vertex v_B in the transcript graph G). It is almost okay to let z be exactly z', if she just received $\mathsf{ECC}(z', ?)$ from Bob so that $z' \in \Sigma^2$ are the last two tree code symbols in the encoding of U_B; however this causes a misalignment in $\ell(v_B)$ and the length of U_A that requires a different convention to fix.

To elaborate, when Alice asks a question, she sends the last two symbols of the tree code at indices $|U_A| - 1$ and $|U_A|$. When she answers Bob's question, she might want to send the symbols at positions $|U_B|$ and $|U_B| - 1$ of what she believes to be Bob's update sequence U_B. However, U_B (which has length $\ell(v_B)$) is shorter than U_A, since it was last updated on the previous message. This clashes with our requirement that when Alice and Bob both have the correct transcript $\mathcal{T}$ as the evaluation of their guesses U_A and U_B, then Bob must interpret the token z in Alice's message as the same regardless of whether she is asking or answering a question. To resolve this, we say that after she decodes Bob's message to v_B, she adds $\bullet\bullet$ onto it; this makes it the same length as U_A, and then she responds with the last two symbols of the new encoding $\mathsf{C}(v_B, \bullet\bullet)$. Additionally, every time she updates U_A, she first updates U_A with $\bullet\bullet$ (as a space holder that says "do nothing"). The result is that both U_A and U_B increase in length by 4 every time the corresponding party receives a message and makes an update. For instance, after Bob has sent the k'th message (so both Alice and Bob have sent $k/2$ messages), Alice updates so that U_A goes from length $2(k-1)$ to length $2(k+1)$, where the first two updates are simply $\bullet\bullet$ and the next two correspond to the additions to U_A. Meanwhile, U_B is of length $2k$, so if she wishes to answer $v_B = v(U_B)$, she would add $\bullet\bullet$ to v_B to make it length $2(k+1)$ as well, and then send the last two symbols in the tree code encoding.

Finally, we discuss a point glossed over so far: how Alice actually decodes Bob's question to v_B if she only receives the encoding of the most recent two symbols $z \in \Sigma^2$ of his transcript guess U_B. She tracks $P_A \in (\Sigma^2)^*$ as a history of all the symbols $\in \Sigma^2$ that she and Bob have sent. That is, every time she sends or receives a message $\mathsf{ECC}(z \in \Sigma^2, \delta)$, she appends z to P_A. Note that P_A has the correct symbols of the tree code encoding of U_B whenever Alice correctly receives Bob's question. Theorem 6.11 says that most of the time when Alice correctly receives Bob's question $\mathsf{ECC}(z, ?)$, she can decode his entire tree code encoding of U_B correctly (even though many elements of P_A do not even correspond to Bob's messages!).

To remember the rules for U_A and P_A, it is helpful to keep in mind the following picture. After Alice speaks in the k'th round, i.e. a total of k messages by either Alice or Bob have been sent so far, both U_A and P_A should be of length $2k$. U_A is of the form $\ldots || \bullet\bullet || (\delta_B\delta_A)_{k-2} || \bullet\bullet || (\delta_B\delta_A)_k$. That is, entries of U_A that are $\bullet\bullet$ are when Bob is talking. Meanwhile, P_A is of the form $\ldots || z_{B,k-3} || z_{A,k-2} || z_{B,k-1} || z_{A,k}$, where $z_{A,i}$ corresponds to the symbols she sent in round i, and $z_{B,i}$ corresponds to the symbols she received in round i.

Partial Corruptions. Lastly, we mention how we handle partial corruptions, i.e. if a received message is not a codeword. The receiver will choose a nearby codeword (with distance $< \frac{1}{3}$ if the codeword is an answer to the party's last question, or with distance $\frac{1}{6} - \epsilon$ if the codeword is a question). With probability proportional to the distance from the codeword, they default to sending a question. Otherwise, they will respond to that codeword as we have described above.

Summary. A brief summary of the most important details:

- Every message Alice sends is of the form $ECC(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow, ?\})$. The instruction δ is ? if Alice is asking Bob a question (potentially also responding to his question), and $0, 1$ or

$\leftarrow$ if she is only responding to his question.

- After receiving a message, Alice performs four updates to both U_A, appending $\bullet\bullet$ and two symbols in $\{0, 1, \leftarrow, \bullet\}$. She similarly performs four updates to P_A, appending the two symbols $z^* \in \Sigma^2$ received in Bob's message and then appending the two symbols z that she is sending in her own next message.
- After sending message k, U_A and P_A are both length $2k$.
- Partial corruptions are handled by performing the behavior described in this section with probability linearly decreasing with the distance to a nearby codeword. The default message is a question.

Indexing: Notational Change. Thus far, we have described U_A and P_A as being a length $2k$ sequence of symbols in $\{0, 1, \leftarrow, \bullet\}$ and Σ respectively, where Alice has just sent the k'th message. Note however that symbols are always appended to U_A and P_A in pairs. Thus, we can instead regard the alphabets of U_A and P_A as being pairs of updates/layered code symbols instead. Throughout the rest of this section, we instead regard $U_A \in (\{0, 1, \leftarrow, \bullet\}^2)^*$ and $P_A \in (\Sigma^2)^*$, so that after Alice sends the k'th message both U_A and P_A are length k. Then, for instance $U_A[k]$ denotes the last two updates Alice has made to U_A, while $U_A[k-1] = \bullet\bullet$.

Similarly, the alphabet of $\mathsf{C}(U_A)$ is Σ^2, so that $\mathsf{C}(U_A)$ is of length $k = |U_A|$. For instance, $\mathsf{C}(U_A)[|U_A|]$ are the last two symbols of $\mathsf{C}(U_A)$.

7.2.1 Formal Description of Protocol

fProtocol 7-1 : Inefficient, Positive Rate Scheme Resilient to $\approx \frac{1}{6}$ Errors

Recall that π_0 is a an alternating, noiseless protocol of length n_0, such that Alice speaks first and her first message is a 1. Alice and Bob have inputs x and y respectively, determining their behavior in this protocol. The noiseless protocol has transcript $\mathcal{T} = \mathcal{T}(x, y) \in \{0, 1\}^{n_0}$. Our error-resilient protocol consists of $K = \frac{n_0}{\epsilon}$ messages numbered $1, \ldots, K$, each consisting of $M(|\Sigma|, \epsilon) = O_\epsilon(1)$ bits. Alice sends the odd messages and Bob sends the even.
Recall that C is an ϵ-sensitive layered code of G with the alphabet Σ. Alice and Bob first (non-interactively) agree on an explicit choice of C by testing each labeling of G and taking the lexicographically first layered code that is ϵ-sensitive.

Alice and Bob track a private sequence of updates of the transcript guess, denoted $U_A, U_B \in \{0, 1, \leftarrow, \bullet\}^2)^*$ respectively initialized to $\emptyset$. They also track confidence weights $w_A, w_B \in \mathbb{N}$, both initialized to 0. Alice and Bob additionally track the sequence $P_A, P_B \in (\Sigma^2)^*$ of pairs of symbols $\in \Sigma^2$ that they have sent and received throughout the protocol. P_A, P_B are both initialized to $\emptyset$.

In what follows, we describe Alice's behavior. Bob's behavior is identical, except notationally switching x and y, and A and B. At the end of the protocol, Alice and Bob output $(t(v(U_A)), \frac{2w_A}{K})$ and $(t(v(U_B)), \frac{2w_A}{K})$ respectively.
Alice's first turn is special; she sets $U_A = \bullet 1$, sets $P_A = \mathsf{C}(\bullet 1)$, and sends $\mathsf{ECC}(\mathsf{C}(\bullet 1), ?)$.

f Alice

Alice has just received a message m from Bob. Let $\mathsf{asked} = \mathsf{true}$ if the last message she sent was of the form $\mathsf{ECC}(z, ?)$ for some $z \in \Sigma^2$ and false otherwise (we let $\mathsf{asked} = \mathsf{false}$ in the first round for Bob). Let $d_m(z, \delta)$ denote $\frac{1}{M(|\Sigma|,\epsilon)} \cdot \Delta(m, \mathsf{ECC}(z, \delta))$.

Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$ and $z_A \in \Sigma^2$ to be $\mathsf{C}(U_A)[|U_A|]$. Then, she picks the first of the following cases that holds.

fCase 1: $\mathsf{asked} = \mathsf{true}$ *and for some* $\delta \in \{0, 1, \leftarrow, ?\}$*, we have* $d_m(z_A, \delta) < \frac{1}{3}$.
Let $p = 1 - 3d_m(z_A, \delta)$.

- Let the instruction $\hat{\delta} = \delta$ unless $\delta = ?$, in which case $\hat{\delta} = 1$. Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \hat{\delta}$ and otherwise (with probability $1 - p$), sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$. She computes $\zeta = \mathsf{C}(U_A)[|U_A|]$.
- Alice sets $P_A \leftarrow P_A||z_A||\zeta$.
- Alice sends $\mathsf{ECC}(\zeta, ?)$.

fCase 2: *For some* $z^* \in \Sigma^2$*, we have* $d_m(z^*, ?) \leq \frac{1}{6} - \epsilon$.
Alice computes $v^* = \mathsf{CDec}(P_A||z^*)$.

fSubcase 2.1: $v^* = \perp$.

- Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$. Alice sets $\zeta = \mathsf{C}(U_A)[|U_A|]$.
- Alice sets $P_A \leftarrow P_A||z^*||\zeta$.
- Alice sends $\mathsf{ECC}(\zeta, ?)$.

In the next two subcases, $v^* \in \Pi$. Let $T^* = t(v^*)$.

fSubcase 2.2: T^* *is complete, i.e.* $|T^*| = n_0$*, and is consistent with* x.
Let $p = 0.5 - 3d_m(z^*, ?)$.

- Alice computes $\hat{\delta} = \mathsf{op}_{T^*}(t(v(U_A)))$. With probability p, Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \hat{\delta}$ and otherwise (with probability $1-p$), sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$. She sets $\zeta = \mathsf{C}(U_A)[|U_A|]$.
- Alice sets $P_A \leftarrow P_A||z^*||\zeta$.
- Alice sends $\mathsf{ECC}(\zeta, ?)$.

fSubcase 2.3: $|T^*| \neq n_0$ *or* T^* *is inconsistent with* x.
Let $p = 1 - 6d_m(z^*, ?)$.

- Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$.
- With probability p, Alice computes $\delta = \mathsf{op}_x(T^*)$ and sends $\mathsf{ECC}(\zeta := \mathsf{C}(v^*, \bullet\bullet), \delta)$. Else (with probability $1 - p$), she sends $\mathsf{ECC}(\zeta := \mathsf{C}(U_A)[|U_A|], ?)$.
- Alice sets $P_A \leftarrow P_A||z^*||\zeta$.

fCase 3: *None of the above.*

- Alice sets $(U_A, w_A) \leftarrow (U_A, w_A) \otimes_x \bullet$. She computes $\zeta = \mathsf{C}(U_A)[|U_A|]$.
- Alice sets $P_A \leftarrow P_A||z||\zeta$, where $z \in \Sigma^2$ is some arbitrary pair of symbols.
- Alice sends $\mathsf{ECC}(\zeta, ?)$.

7.3 Main Theorems

Theorem 7.6. *Protocol 7-1 is a* $\left(\frac{1}{6}, 1224\epsilon, 2 \cdot \exp\left(-\frac{\epsilon n_0}{800}\right)\right)$*-scaling scheme with communication complexity* $O_\epsilon(n_0)$ *and computational complexity* $2^{2^{O_\epsilon(n_0)}}$.

We prove Theorem 7.6 in Section 7.4. Combining Theorem 7.6 with the boosting procedure in

Protocol 5-1, we obtain the following result.

Corollary 7.7. *For any $\epsilon > 0$ there is a scheme for noiseless protocols of length n_0 that is resilient to $\left(\frac{1}{6} - \epsilon\right)$-fraction of errors with probability $1 - e^{-\epsilon n_0/40 \log^4 n_0}$. The scheme has communication complexity $O_\epsilon(n_0)$ and computational complexity $\tilde{O}_\epsilon(n_0)$.*

Proof. Let $\epsilon' = \epsilon/256$, and let C_ϵ be such that $e^{-\epsilon' C_\epsilon/10 \log^4 C_\epsilon} < \epsilon'$. We choose $C_\epsilon \geq \frac{8 \cdot 800 \cdot 1224}{\epsilon'^2}$ so that $C_\epsilon \geq \frac{100}{\epsilon'} + 1$ and $\frac{\epsilon'}{4} > 2 \cdot \exp(-\frac{8}{\epsilon'} \cdot \log^4 n_0) \geq 2 \cdot \exp(-\frac{\epsilon' C_\epsilon \log^4 n_0}{800 \cdot 1224})$.
We recursively apply Theorem 5.3 three times.

- We begin with the $(\frac{1}{6}, \epsilon', 2 \cdot \exp(-\frac{\epsilon' n_0}{800 \cdot 1224})$-scaling scheme from Theorem 7.6, which has communication complexity $O_\epsilon(n_0)$ and computational complexity $\exp(\exp_\epsilon(n_0))$.
- Since $2 \cdot \exp(-\frac{\epsilon' C_\epsilon \log^4 n_0}{800 \cdot 1224}) < \frac{\epsilon'}{4}$, we apply Theorem 5.3 to obtain a $(\frac{1}{6}, 4\epsilon', e^{-\epsilon' n_0/10 \log^4 n_0})$-scaling scheme with communication complexity $\frac{n_0}{\epsilon' \log^4 n_0} \cdot O_{\epsilon'}(C_\epsilon \log^4 n_0) = O_\epsilon(n_0)$ and computational complexity $\tilde{O}_{\epsilon'}(n_0) \cdot \exp(\exp_\epsilon(C_\epsilon \log^4 n_0)) = \exp(\exp_\epsilon(\text{polylog} n_0))$. Let $\mu'_{\epsilon'}(n_0) = e^{-\epsilon' n_0/10 \log^4 n_0}$.
- Next, since $\mu'_{\epsilon'}(C_\epsilon \log^4 n_0) = \exp(-\frac{\epsilon' C_\epsilon \log^4 n_0}{10 \log^4 (C_\epsilon \log^4 n_0)}) \leq \exp(-\frac{\epsilon' C_\epsilon}{10 \log^4 C_\epsilon}) < \epsilon' = \frac{4\epsilon'}{4}$, we can apply Theorem 5.3 again to obtain a $(\frac{1}{6}, 16\epsilon', e^{-2\epsilon' n_0/5 \log^4 n_0})$-scaling scheme with communication complexity $\frac{n_0}{4\epsilon' \log^4 n_0} \cdot O_\epsilon(C_\epsilon \log^4 n_0) = O_\epsilon(n_0)$ and computational complexity $\tilde{O}_{\epsilon'}(n_0) \cdot \exp(\exp_\epsilon(\text{polylog}(C_\epsilon \log^4 n_0))) = \exp(\exp_{\epsilon'}(\text{poly}(\log \log(n_0))))$. Let $\mu''_{\epsilon'}(n_0) = e^{-2\epsilon' n_0/5 \log^4 n_0}$.
- Again, since $\mu''_{\epsilon'}(C_\epsilon \log^4 n_0) = \exp(-\frac{2\epsilon' C_\epsilon \log^4 n_0}{5 \log^4 (C_\epsilon \log^4 n_0)}) \leq \exp(-\frac{2\epsilon' C_\epsilon}{5 \log^4 C_\epsilon}) < \epsilon'^4 < \frac{16\epsilon'}{4}$, we can apply Theorem 5.3 to get a $(\frac{1}{6}, 64\epsilon', e^{-8\epsilon' n_0/5 \log^4 n_0})$-scaling scheme with communication complexity $\frac{n_0}{16\epsilon' \log^4 n_0} \cdot O_\epsilon(C_\epsilon \log^4 n_0) = O_\epsilon(n_0)$ and computational complexity $\tilde{O}_\epsilon(n_0) \cdot \exp(\exp_\epsilon(\text{poly}(\log \log(C_\epsilon \log^4 n_0)$ $\exp(\exp_\epsilon(\text{poly}(\log \log \log n_0))) \leq \text{poly}_\epsilon(n_0)$. Let $\mu'''_{\epsilon'}(n_0) = e^{-8\epsilon' n_0/5 \log^4 n_0}$.
- Finally, to further reduce the computational complexity to $\tilde{O}_\epsilon(n_0)$, we apply Theorem 5.3 one last time. Since $\mu'''_{\epsilon'}(C_\epsilon \log^4 n_0) = \exp(-\frac{8\epsilon' C_\epsilon \log^4 n_0}{5 \log^4 (C_\epsilon \log^4 n_0)}) \leq \exp(-\frac{8\epsilon' C_\epsilon}{5 \log^4 C_\epsilon}) < \epsilon'^{16} < \frac{64\epsilon'}{4}$, we get a $(\frac{1}{6}, 256\epsilon', e^{-32\epsilon' n_0/5 \log^4 n_0})$-scaling scheme with communication complexity $\frac{n_0}{64\epsilon' \log^4 n_0} \cdot O_\epsilon(C_\epsilon \log^4 n_0) = O_\epsilon(n_0)$ and computational complexity $\tilde{O}_\epsilon(n_0) \cdot \text{poly}_\epsilon(C_\epsilon \log^4 n_0) = \tilde{O}_\epsilon(n_0)$.

Thus, we have arrived at a $(\frac{1}{6}, \epsilon, e^{-\epsilon n_0/40 \log^4 n_0})$-scaling scheme.

□

7.4 Analysis

Note that Alice and Bob only ever append to U_A, U_B, P_A, P_B, and once a symbol has been appended it is never modified. Thus, throughout the analysis, when we refer to U_A, U_B, P_A, P_B, we mean their values at the end of the protocol, so that $U_A, U_B \in (\{0, 1, \leftarrow, \bullet\}^2)^K$ and $P_A, P_B \in (\Sigma^2)^K$.

7.4.1 Unique Decoding Lemma

Definition 7.8 ($\mathcal{S}$)**.** *We define the set $\mathcal{S}$ to consist of all rounds $k \in [K]$ where one of the following conditions does* not *hold.*

(i) For not necessarily distinct parties $P, P' \in \{A, B\}$, it holds that $\mathsf{C}(U_P)[k] = P_{P'}[k] \in \Sigma^2 \implies \mathsf{CDec}(P_{P'}[1:k]) = v(U_P[1:k])$.

(ii) $\mathsf{C}(U_A)[k] = \mathsf{C}(U_B)[k] \in \Sigma^2 \implies v(U_A[1:k]) = v(U_B[1:k])$.

Lemma 7.9. *$\mathcal{S}$ has size at most $20\epsilon K$.*

Proof. We deal with each of the conditions individually.

(i) Let $\mathcal{S}_1$ be the set of indices that violate the first condition. For each pair of parties P, P', by Theorem 6.11, it holds that there are only $2\epsilon \cdot 2K$ values of k where $\mathsf{C}(U_P)[k] = P_{P'}[k] \implies \mathsf{C}(U_P)[k][2] = P_{P'}[k][2]$,[3] but $\mathsf{CDec}(P_{P'}[1:k]) \neq v(U_P[1:k])$. Thus, adding over all four cases of $P, P' \in \{A, B\}$, it holds that $\mathcal{S}_1$ has size at most $4 \cdot 2\epsilon 2K = 16\epsilon K$.

(ii) Let $\mathcal{S}_2$ be the set of indices that violate the second condition. By Theorem 6.11, it holds that there are only $2\epsilon \cdot 2K$ values of k where $\mathsf{C}(U_A)[k] = \mathsf{C}(U_B)[k] \implies \mathsf{C}(U_A)[k][2] = \mathsf{C}(U_B)[k][2]$ but $v(U_A[1:k]) \neq \mathsf{CDec}(\mathsf{C}(U_B[1:k]))$. The latter is always either $v(U_B[1:k])$ or $\perp$, so there are at most $2\epsilon \cdot 2K$ values of k where $v(U_A[1:k]) \neq v(U_B[1:k])$. Thus, $\mathcal{S}_2$ is size at most $4\epsilon K$.

The total size of $\mathcal{S}$ is at most $|\mathcal{S}_1| + |\mathcal{S}_2| \leq 20\epsilon K$. □

7.4.2 Definitions for the Potential

To prove Theorem 7.6, we analyze the effects of corruption on the *good* and *bad updates* Alice/Bob make. We begin by defining good, bad, and neutral updates. After receiving a message from Bob, Alice updates her transcript U_A and confidence w_A to U'_A and w'_A.

- Let $(\mathcal{U}'_A, \mathcal{W}'_A) = (U_A, w_A) \otimes_x \mathsf{op}_{\mathcal{T}}(t(v(U_A)))$. The update is good if $t(v(\mathcal{U}'_A)) = t(v(U'))$ and $\mathcal{W}'_A = w_A$.
- The update is neutral if $(t(v(U'_A)), w'_A) = (t(v(U_A)), w_A)$.
- The update is bad otherwise.

We similarly define good and bad updates for Bob. We will often refer to making a good/bad update as simply *making an update*, and considering a neutral update as having done nothing.

For each $t \in [1, \ldots, K]$, 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 $1, \ldots, t$. Note that she only updates in response to messages she receives (the even numbered messages).
- 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 $1, \ldots, t$. Note that he only updates in response to messages he receives (the odd numbered messages).

Lemma 7.10. *The potential ψ_t^A determines Alice's final transcript guess and her confidence as follows:*

[3] Recall that $\mathsf{C}(U_P)[k], P_{P'}[k] \in \Sigma^2$ so $\mathsf{C}(U_P)[k][2], P_{P'}[k][2] \in \Sigma$.

(i) If $\psi_t^A \geq n_0/2$, then $t(v(U_A)) = \mathcal{T}$ and $w_A \geq \psi_t^A - n_0/2$.
(ii) If $\psi_t^A \leq n_0/2$, then $t(v(U_A)) \neq \mathcal{T}$ and $w_A \leq n_0/2 - \psi_t^A$.

The same statements hold for Bob, replacing A with B.

Proof. We prove this for Alice as the proof for Bob is identical. After sending message 1, since $U_A = \bullet 1$, in order make $t(v(U_A)) = \mathcal{T}$, Alice needs to perform $n_0/2$ good updates (the first $n_0/2 - 1$ updates consist of appending two bits, corresponding to Bob's and her next messages in π_0, followed by 1 further good update consisting of simply appending Bob's next message). Every good update thereafter increases w_A by 1 without changing $t(v(U_A))$.

It remains to show that every good update undoes a bad update; that is, every bad update, when followed by a good update, results back in the original value of $(t(v(U_A)), w_A)$. If the bad update appends two instructions $\in \{0, 1, \bullet\}^2 \backslash \{\bullet\bullet\}$ to U_A, then the new value of $t(v(U_A))$ must not be a prefix of $\mathcal{T}$. Then the next good instruction, which is $\leftarrow$, undoes this. If the bad update deletes the last one or two bits of $t(v(U_A))$ incorrectly, then re-appending the bit(s) undoes this. If the bad update increases w_A incorrectly, then $t(v(U_A)) \neq \mathcal{T}$, so the next good update is $\mathsf{op}_{\mathcal{T}}(t(v(U_A)))$ which causes w_A to decrease by 1. If the bad update decreases w_A incorrectly, then $t(v(U_A)) = \mathcal{T}$, and the next good update is $\mathsf{op}_{\mathcal{T}}(t(v(U_A)))$ which increases w_A by 1. □

From this point on, we will focus on analyzing Protocol 7-1 from Alice's perspective, as the analysis from Bob's perspective follows analogously.

Define ρ_t^A as follows (and similarly ρ_t^B): 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 the protocol so far, if message t is uncorrupted. (Note that $\rho_t^A = 0$ for odd t since Alice sends the odd messages.)

Define val_t^A as follows:

$$\mathsf{val}_t^A = \begin{cases} 0.5 & \text{if } t \text{ is odd and message } t \text{ is of the form } \mathsf{ECC}(z \in \Sigma^2, ?) \text{ and } (\psi_t^A < n_0/2 \text{ or } \psi_{t-1}^B \geq n_0/2). \\ 0.5 & \text{if } t \text{ is even and message } t \text{ is of the form } \mathsf{ECC}(z \in \Sigma^2, ?) \text{ and } \psi_t^B < n_0/2. \\ 0 & \text{otherwise} \end{cases}$$

Define the potential Ψ_t^A as follows:

$$\Psi_t^A = \psi_t^A + \rho_{t+1}^A + \min(\psi_t^B + \rho_{t+1}^B, n_0/2) + \mathsf{val}_{t+1}^A$$

Finally, we define Alice's actual update: Λ_t^A is the actual value of the update Alice makes in response to message t (in particular, $\Lambda_t^A \in \{-1, 0, 1\}$).

Throughout the analysis, we say Alice *interprets* a message m as $\mathsf{ECC}(z^*, \delta^*)$ in Protocol 7-1 when she enters Case 1 or Case 2 according to that value. Additionally, we will say she interprets the message correctly or incorrectly, if $\mathsf{ECC}(z^*, \delta^*)$ respectively equals or does not equal the message Bob sent.

Lemma 7.11. *The following are true for any $k \notin \mathcal{S}$:*

1. *$\rho_k^A \geq 0$. As a corollary, if Alice correctly interprets message k, then $\Lambda_k^A \geq 0$.*
2. *For any k, it holds that $\mathbb{E}[\Lambda_k^A] - \rho_k^A \geq -3\alpha_k - 3\epsilon$.*

3. *For all even k, if Alice interprets message k incorrectly, then $\mathbb{E}[\Lambda_k^A] \geq 0.5 - 3\alpha_k - 3\epsilon$. Similarly, for all odd k, if Bob interprets message k incorrectly, then $\mathbb{E}[\Lambda_k^B] \geq 0.5 - 3\alpha_k - 3\epsilon$.*
4. *Whenever Alice sends $\mathsf{ECC}(z \in \Sigma^2, ?)$ as message k, it holds that $\mathsf{val}_k^A + \rho_k^B \geq 0.5$.*
5. *Whenever Bob sends $\mathsf{ECC}(z \in \Sigma^2, ?)$ as message k, it holds that $\mathsf{val}_k^A + \rho_k^A \geq 0.5$.*

Proof. We prove the statements individually.

1. We assume Alice interprets Bob's message in the k'th round correctly. Let Bob's intended message be $\mathsf{ECC}(z, \delta)$. If $\delta =?$, then $z = \mathsf{C}(U_B)[k]$. We have $z = \mathsf{C}(U_B)[k] = P_A[k]$, so by Lemma 7.9, $v(U_B[1:k]) = \mathsf{CDec}(P_A[1:k])$. Then, if Alice enters Case 1, $\mathsf{CDec}(P_A[1:k]) = v(U_A[1:k])$ as well, so $v(U_A[1:k]) = v(U_B[1:k])$. Since they are the same, they must be either $\emptyset$ or $\mathcal{T}$. In either case, $\hat{\delta} = 1$ results in a positive update. If Alice enters Case 2, then in order to have made an update, she must enter Case 2 Subcase 2, which she only enters if $v(U_B[1:k])$ is complete and consistent with her input, and therefore $= \mathcal{T}$, resulting in a positive update.

   If $\delta \in \{0, 1, \leftarrow\}$, Bob sent $\mathsf{ECC}(P_B[k], \delta)$. The only way that Alice can make an update is by entering Case 1. This requires $P_B[k] = C(U_A[1:k])[k] \implies \mathsf{CDec}(P_B[1:k]) = v(U_A[1:k])$. Note also that Bob must have decoded $\mathsf{CDec}(P_B[1:k-1])$ to v^* and set $P_B[k] \leftarrow \mathsf{C}(v^*, \bullet\bullet)$. Then, $\mathsf{CDec}(P_B[1:k]) \in \{v^* \oplus \bullet \oplus \bullet, \perp\}$. Since $\mathsf{CDec}(P_B[1:k]) = v(U_A[1:k]) \neq \perp$, it holds that $\mathsf{CDec}(P_B[1:k]) = v^* \oplus \bullet \oplus \bullet \implies v(U_A[1:k]) = v^* \oplus \bullet \oplus \bullet)$. This means that Bob sends an instruction which causes Alice to make a positive update.

   To show $\Lambda_k^A \geq 0$, Alice either makes the update corresponding to the case she is in, or no update at all. In order for $\rho_k^A \geq 0$, this one possible update she could make must be a good update, so $\Lambda_k^A \geq 0$ as well.
2. Clearly, if k is odd, then $\Lambda_k^A - \rho_k^A = 0 \geq -3\alpha_k - 3\epsilon$. We focus on when k is even. Let Bob's intended message be $\mathsf{ECC}(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow, \bullet\})$

   We split the proof into cases.

   Case 1: Alice does not enter Case 1 or Case 2 Subcase 2.

   Alice does not update, so $\Lambda_k^A = 0$. If Bob's message was of the form $\mathsf{ECC}(z_A, \delta)$, then $\rho_k^A \leq 1$ and $\alpha_k \geq \frac{1}{3}$ (otherwise Alice should have entered Case 1). This gives

$$\begin{aligned} &\mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & 0 - 1 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

   Otherwise if Bob's message was of the form $\mathsf{ECC}(z^* \neq z_A \in \Sigma^2, ?)$, then $\alpha_k \geq \frac{1}{6} - \epsilon$. He must enter Case 2 or Case 3, so his expected update is at most 0.5. Then,

$$\begin{aligned} &\mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & 0 - 0.5 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

   Case 2: Alice interprets message k correctly and she enters Case 1 or Case 2.

   We have $d_m \leq \alpha_k$. We only need to look at the case where her possible update

is positive; if it is 0, the result follows from the calculation above and cannot be negative. If she enters Case 1, her probability of updating is $1 - 3d_m \geq 1 - 3\alpha_k$, so

$$\begin{aligned} & \mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & (1 - 3\alpha_k) - 1 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

If she enters Case 2 Subcase 2, her probability of updating is $0.5 - 3d_m \geq 0.5 - 3\alpha_k$, so

$$\begin{aligned} & \mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & (0.5 - 3\alpha_k) - 0.5 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

Case 3: Alice interprets message k incorrectly as $\mathsf{ECC}(z^*, \delta^*)$ and enters Case 1 or Case 2 Subcase 2.

If she enters Case 1 and $z = z^*$, then $d_m \geq \frac{2}{3} - \alpha_k$ so her probability of updating is $1 - 3d_m \leq -1 + 3\alpha_k$, so

$$\begin{aligned} & \mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & -1(-1 + 3\alpha_k) - 1 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

If she enters Case 1 and $z \neq z^*$, then $d_m \geq \frac{1}{2} - \epsilon - \alpha_k$, so her probability of updating is $1 - 3d_m \leq -0.5 + 3\alpha_k + 3\epsilon$. Also, $\rho_k^A \leq 0.5$. This gives

$$\begin{aligned} & \mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & -1(-0.5 + 3\alpha_k + 3\epsilon) - 0.5 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

If she enters Case 2 Subcase 2, then $d_m \geq \frac{1}{2} - \epsilon - \alpha_k$, so her probability of updating is $0.5 - 3d_m \leq -1 + 3\alpha_k + 3\epsilon$. This gives

$$\begin{aligned} & \mathbb{E}[\Lambda_k^A] - \rho_k^A \\ \geq\ & -1(-1 + 3\alpha_k + 3\epsilon) - 1 \\ \geq\ & -3\alpha_k - 3\epsilon. \end{aligned}$$

3. We prove this for Alice as the proof for Bob is symmetric. If $\rho_k^A \geq 0.5$, then the result follows from the previous item. Otherwise, $\rho_k^A = 0$. Alice interprets message k as (z^*, δ^*) and Bob's intended message was (z, δ), where $(z^*, \delta^*) \neq (z, \delta)$.

   If she enters Case 1 and $z = z^*$, then $d_m \geq \frac{2}{3} - \alpha_k$ so her probability of updating is $1 - 3d_m \leq$

$-1 + 3\alpha_k$, so

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A]\\ \geq\ &-1(-1+3\alpha_k)\\ \geq\ &1-3\alpha_k-3\epsilon.\end{aligned}$$

If she enters Case 1 and $z \neq z^*$, then $d_m \geq \frac{1}{2} - \epsilon - \alpha_k$, so her probability of updating is $1 - 3d_m \leq -0.5 + 3\alpha_k + 3\epsilon$. Also, $\rho_k^A \leq 0.5$. This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A]\\ \geq\ &-1(-0.5+3\alpha_k+3\epsilon)\\ \geq\ &0.5-3\alpha_k-3\epsilon.\end{aligned}$$

If she enters Case 2 Subcase 2, then $d_m \geq \frac{1}{2} - \epsilon - \alpha_k$, so her probability of updating is $0.5 - 3d_m \leq -1 + 3\alpha_k + 3\epsilon$. This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A]\\ \geq\ &-1(-1+3\alpha_k+3\epsilon)\\ \geq\ &1-3\alpha_k-3\epsilon.\end{aligned}$$

4. Alice sends the odd messages, so we are in the case where k is odd. If $\psi_t^A < n_0/2$ or $\psi_{t-1}^B \geq n_0/2$, then the result follows because $\mathsf{val}_k^A = 0.5$ and $\rho_k^B \geq 0$. Otherwise $\psi_k^A = \psi_{k-1}^A \geq n_0/2$. Thus, Alice's message is $\mathsf{ECC}(\mathsf{C}(U_A)[k], ?)$ where $t(v(U_A)) = \mathcal{T}$. If Bob receives this message uncorrupted, then $\mathsf{C}(U_A)[k] = P_B[k]$, so by Definition 7.8, $v(U_A[1:k]) = \mathsf{CDec}(P_B[1:k])$. If he enters Case 1, then $\mathsf{C}(U_A)[k] = \mathsf{C}(U_B)[k] \implies \mathcal{T} = t(v(U_A[1:k])) = t(v(U_B[1:k]))$ so it must be the case that he makes a good update. If he enters Case 2, he decodes v^* such that $t(v^*) = \mathcal{T}$, and so also makes a good update with at least 0.5 probability.
5. The proof is very similar. Bob sends the odd messages, so we are in the case where k is even. If $\psi_t^B < n_0/2$, then the result follows because $\psi_k^A = 0.5$ and $\rho_k^A \geq 0$. Otherwise $\psi_k^B = \psi_{k-1}^B \geq n_0/2$. Thus, Bob's message is $\mathsf{ECC}(\mathsf{C}(U_B)[k], ?)$ where $t(v(U_B)) = \mathcal{T}$. If Alice receives this message uncorrupted, then $\mathsf{C}(U_B)[k] = P_A[k]$, so by Definition 7.8, $v(U_B[1:k]) = \mathsf{CDec}(P_A[1:k])$. If she enters Case 1, she makes a good update, and if she enters Case 2, she decodes v^* such that $t(v^*) = \mathcal{T}$, and so also makes a good update with at least 0.5 probability.

□

7.4.3 Calculating the Change in Potential

The main objective is to prove the following lemma.

Lemma 7.12. *For any $k \in [K]$ such that $k-1, k, k+1 \notin \mathcal{S}$, if an α_k fraction of message k is corrupted, then*

$$\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \geq 0.5 - 3\epsilon - 3\alpha_k.$$

Proof. We split the proof into four parts depending on the parity of k and on the value of ψ_k^B or ψ_{k-1}^B.

k **is even and** $\psi_k^B < n_0/2$**.** Then

$$\begin{aligned}
&\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \\
&= \mathbb{E}[\psi_k^A + \rho_{k+1}^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) + \mathsf{val}_{k+1}^A - \psi_{k-1}^A - \rho_k^A - \min(\psi_{k-1}^B + \rho_k^B, n_0/2) - \mathsf{val}_k^A] \\
&= \mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) - \min(\psi_k^B, n_0/2)] \\
&= \mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \rho_{k+1}^B].
\end{aligned}$$

Case 1: Message k is of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$.

Notice that $z = \mathsf{C}(U_B)[k]$.

It holds that $\mathsf{val}_k^A = 0.5$. If the message is uncorrupted, Alice must enter Case 2 Subcase 3 because $\mathsf{CDec}(P_A[1:k]) = v(U_B[1:k]) \neq v(U_A[1:k])$ by Definition 7.8. Alice only enters Case 1 when $\mathsf{CDec}(P_A[1:k]) = v(U_A[1:k])$. Thus, $\rho_k^A = 0$ because Alice makes a neutral update. Thus, we need to show

$$\mathbb{E}[\Lambda_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B] \geq 1 - 3\alpha_k - 3\epsilon.$$

Subcase 1.1: Alice interprets message k correctly.

Then we must be in Case 2 Subcase 3 as shown earlier. Also, $\Lambda_k^A = 0$. With probability at least $1 - 6\alpha_k$, Alice sends a message of the form $\mathsf{ECC}(\mathsf{C}(U_B)[k+1], \delta)$ upon computing $\mathsf{CDec}(P_A[1:k]) = v(U_B[1:k])$. This results in $\rho_{k+1}^B = 1$. Otherwise (with probability at most $6\alpha_k$), she sends $\mathsf{ECC}(\mathsf{C}(U_A)[k+1], ?)$; then by Lemma 7.11 $\mathsf{val}_{k+1}^A + \rho_{k+1}^A \geq 0.5$. Overall, this evaluates to

$$\begin{aligned}
&\mathbb{E}[\Lambda_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B] \\
&= 0 + (1 - 6\alpha_k)(1) + 6\alpha_k(0.5) \\
&= 1 - 6\alpha_k + 3\alpha \\
&\geq 1 - 3\alpha_k - 3\epsilon.
\end{aligned}$$

Subcase 1.2: Alice enters Case 3.

$\Lambda_k^A = 0$ and $\mathsf{val}_{k+1}^A + \rho_{k+1}^B \geq 0.5$ by Lemma 7.11. Also, $\alpha_k \geq \frac{1}{6} - \epsilon$. This gives

$$\begin{aligned}
&\mathbb{E}[\Lambda_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B] \\
&= 0 + 0.5 \\
&\geq 1 - 3\alpha_k - 3\epsilon.
\end{aligned}$$

Subcase 1.3: Alice interprets message k incorrectly as $\mathsf{ECC}(z_A, \delta \in \{0, 1, \leftarrow, \delta\})$.

We have $\mathbb{E}[\Lambda_k^A] \geq 0.5 - 3\alpha_k - 3\epsilon$ by Lemma 7.11 regardless of whether $z_A = z$. Alice sends a message of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$ so $\mathsf{val}_{k+1}^A + \rho_{k+1}^B \geq 0.5$ by Lemma 7.11.

This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B]\\ &= 0.5 - 3\alpha_k - 3\epsilon + 0.5\\ &\geq 1 - 3\alpha_k - 3\epsilon.\end{aligned}$$

Subcase 1.4: Alice interprets message k incorrectly as (z^*, δ) where $z^* \neq z_A$.

Let d_m be the relative distance from the received message to $\mathsf{ECC}(z^*, \delta)$. Notice that Alice updates with probability $0.5 - 3d_m \leq 0.5 - 3(0.5 - \epsilon - \alpha_k) = -1 + 3\alpha_k + 3\epsilon$ probability, so

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B]\\ &\geq \Lambda_k^A\\ &\geq -1(-1 + 3\alpha + 3\epsilon)\\ &\geq 1 - 3\alpha_k - 3\epsilon.\end{aligned}$$

Case 2: Message k is of the form $\mathsf{ECC}(z, \delta)$ for some $\delta \in \{0, 1, \leftarrow\}$.

We have $\mathsf{val}_k^A = 0$ because $\delta \neq ?$. Thus, we need to show

$$\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B] \geq 0.5 - 3\alpha_k - 3\epsilon.$$

Subcase 2.1: Alice enters any case except Case 2 Subcase 3.

We have $\mathbb{E}[\Lambda_k^A] - \rho_k^A \geq -3\alpha_k - 3\epsilon$ by Lemma 7.11 and $\mathsf{val}_{k+1}^A + \rho_{k+1}^B \geq 0.5$ by Lemma 7.11. This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B]\\ &\geq -3\alpha_k - 3\epsilon + 0.5\\ &= 0.5 - 3\alpha_k - 3\epsilon.\end{aligned}$$

Subcase 2.2: Alice enters Case 2 Subcase 3.

$\Lambda_k^A = 0$ because Alice does not update. Also $\rho_k^A \leq 1$. Alice must have interpreted incorrectly since the received message has $\delta = ?$, so with probability of at least $1 - p \geq 6(0.5 - \epsilon - \alpha_k)$, Alice sends a message of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$, where $\mathsf{val}_{k+1}^A + \rho_{k+1}^B \geq 0.5$. This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A + \rho_{k+1}^B]\\ &\geq 0 - 1 + 6(0.5 - \epsilon - \alpha_k) \cdot 0.5 + =\\ &\geq 0.5 - 3\alpha_k - 3\epsilon.\end{aligned}$$

k **is even and** $\psi_k^B \geq n_0/2$**.** Then

$$\begin{aligned}
&\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \\
&= \mathbb{E}[\psi_k^A + \rho_{k+1}^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) + \mathsf{val}_{k+1}^A - \psi_{k-1}^A - \rho_k^A - \min(\psi_{k-1}^B + \rho_k^B, n_0/2) - \mathsf{val}_k^A] \\
&= \mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) - \min(\psi_k^B, n_0/2)] \\
&= \mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A].
\end{aligned}$$

We have that $\mathsf{val}_k^A = 0$ because either the message is $\mathsf{ECC}(z \in \Sigma^2, ?)$ with $\psi_k^B \geq n_0/2$, or $\mathsf{ECC}(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow\})$. Thus, we need to show

$$\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A] \geq 0.5 - 3\alpha_k - 3\epsilon.$$

Case 1: Alice does not enter Case 2 Subcase 3.

We know $\Lambda_k^A - \rho_k^A \geq -3\alpha_k - 3\epsilon$ by Lemma 7.11 and $\mathsf{val}_{k+1}^A = 0.5$ because message $k+1$ is of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$. Then

$$\begin{aligned}
&\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A] \\
&\geq -3\alpha_k - 3\epsilon + 0.5 \\
&\geq 0.5 - 3\alpha_k - 3\epsilon.
\end{aligned}$$

Case 2: Alice interprets message k enters correctly and enters Case 2 Subcase 3.

Bob must have sent $\mathsf{ECC}(\mathsf{C}(U_B)[k], ?)$. It holds that $P_A[k] = \mathsf{C}(U_B)[k]$ so by Definition 7.8, unless $k \in \mathcal{S}$, $\mathsf{CDec}(P_A) = v(U_B)$. However, since $\psi_k^B \geq n_0/2$ she must have actually entered Case 2 Subcase 2, which is a contradiction.

Case 3: Alice interprets message k incorrectly and enters Case 2 Subcase 3.

$\Lambda_k^A = 0$ because Alice does not update. Also, $\rho_k^A \leq 1$. With probability at least $1 - p \geq 6(0.5 - \alpha_k)$, Alice sends a message of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$, so $\mathsf{val}_{k+1}^A + \rho_{k+1}^B \geq 0.5$. This gives

$$\begin{aligned}
&\mathbb{E}[\Lambda_k^A - \rho_k^A + \mathsf{val}_{k+1}^A] \\
&\geq 0 - 1 + 6(0.5 - \alpha_k) \cdot 0.5 \\
&\geq 0.5 - 3\alpha_k - 3\epsilon.
\end{aligned}$$

k **is odd and** $\psi_{k-1}^B < n_0/2$**.** Then the expression simplifies to

$$\begin{aligned}
&\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A] \\
&= \mathbb{E}[\psi_k^A + \rho_{k+1}^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) + \mathsf{val}_{k+1}^A - \psi_{k-1}^A - \rho_k^A - \min(\psi_{k-1}^B + \rho_k^B, n_0/2) - \mathsf{val}_k^A] \\
&= \mathbb{E}[\rho_{k+1}^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \min(\psi_k^B, n_0/2) - \min(\psi_{k-1}^B + \rho_k^B, n_0/2)] \\
&= \mathbb{E}[\rho_{k+1}^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \psi_k^B - \psi_{k-1}^B - \rho_k^B] \\
&= \mathbb{E}[\rho_{k+1}^A + \Lambda_k^B - \rho_k^B + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A].
\end{aligned}$$

Case 1: $\psi_k^A \geq n_0/2$ or message k is of the form $\mathsf{ECC}(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow\})$.

We know that $\mathsf{val}_k^A = 0$. Thus, we want to show

$$\mathbb{E}[\Lambda_k^B - \rho_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A] \geq 0.5 - 3\alpha_k - 3\epsilon.$$

Subcase 1.1: Bob does not enter Case 2 Subcase 3.

Bob's next message is of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$ so $\rho_{k+1}^A + \mathsf{val}_{k+1}^A \geq 0.5$ by Lemma 7.11. By the same lemma, $\mathbb{E}[\Lambda_k^B] - \rho_k^B \geq -3\alpha_k - 3\epsilon$. This gives

$$\begin{aligned} &\mathbb{E}[\Lambda_k^B - \rho_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A] \\ &\geq 0.5 - 3\alpha_k - 3\epsilon. \end{aligned}$$

Subcase 1.2: Bob interprets message k correctly and enters Case 2 Subcase 3.

If message k is of the form $\mathsf{ECC}(z \in \Sigma^2, \delta)$ for some $\delta \neq ?$, Bob cannot have entered Case 2. Thus, $\psi_k^A \geq n_0/2$ and Alice must have sent $\mathsf{ECC}(\mathsf{C}(U_A)[k], ?)$, and so $P_B[k] = \mathsf{C}(U_A)[k]$. Then by Definition 7.8, $\mathsf{CDec}(P_B[1:k]) = v(U_A[1:k])$, and since $\psi_k^A \geq n_0/2$, it holds that $t(\mathsf{CDec}(P_A[1:k])) = t(v(U_A)) = \mathcal{T}$. Then, Bob enters Case 2 Subcase 2, which is a contradiction.

Subcase 1.3: Bob interprets message k incorrectly and enters Case 2 Subcase 3.

$\Lambda_k^B = 0$ and Bob sends $\mathsf{ECC}(z \in \Sigma^2, ?)$ with probability $1 - p \geq 6(0.5 - \epsilon - \alpha_k)$ resulting in $\mathsf{val}_{k+1}^A + \rho_{k+1}^A \geq 0.5$, so

$$\begin{aligned} &\mathbb{E}[\Lambda_k^B - \rho_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A] \\ &\geq 0 - 1 + 0.5(3 - 6\epsilon - 6\alpha_k) \\ &= 0.5 - 3\alpha_k - 3\epsilon. \end{aligned}$$

Case 2: Message k is of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$ and $\psi_k^A < n_0/2$.

Note that $z = \mathsf{C}(U_B)[k]$ and we know that $\mathsf{val}_k^A = 0.5$ and $\rho_k^B = 0$. Thus, we need to show

$$\mathbb{E}[\Lambda_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A] \geq 1 - 3\alpha_k - 3\epsilon.$$

Subcase 2.1: Bob interprets message k correctly.

Bob must enter Case 2 Subcase 3. This is because $v(U_B[1:k]) \neq v(U_A[1:k])$, so Bob cannot enter Case 1 by Definition 7.8. Upon entering Case 2, he correctly decodes $\mathsf{CDec}(P_B[1:k]) = v(U_A[1:k])$, causing him to enter Case 2 Subcase 3. Then, with $p \geq 1 - 6\alpha_k$, we have $\rho_{k+1}^A = 1$, because Bob sends $\mathsf{ECC}(\mathsf{C}(U_A)[k+1], \delta)$, where δ is such that Alice would make a positive update upon entering Case 1 if she interprets the message correctly. Otherwise $\rho_{k+1}^A + \mathsf{val}_{k+1}^A \geq 0.5$. By Lemma 7.11,

$\Lambda_k^B \geq 0$, which gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A]\\ &\geq 1(1-6\alpha_k) + 0.5(6\alpha_k) + 0\\ &\geq 1 - 3\alpha_k - 3\epsilon.\end{aligned}$$

Subcase 2.2: Bob interprets message k incorrectly and does not enter Case 2 Subcase 3.

Notice $\Lambda_k^B > 0.5 - 3\alpha_k - 3\epsilon$ by Lemma 7.11, and $\rho_{k+1}^A + \mathsf{val}_{k+1}^A \geq 0.5$ by Lemma 7.11 since he sends $\mathsf{ECC}(z \in \Sigma^2, ?)$ in all cases except Case 2 Subcase 3. This gives

$$\begin{aligned}&\mathbb{E}[\Lambda_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A]\\ &\geq 0.5 - 3\alpha_k - 3\epsilon + 0.5\\ &\geq 1 - 3\alpha_k - 3\epsilon.\end{aligned}$$

Subcase 2.3: Bob interprets message k incorrectly and enters Case 2 Subcase 3.

Notice $\Lambda_k^B = 0$ and $\alpha_k \geq \frac{1}{3}$.

$$\begin{aligned}&\mathbb{E}[\Lambda_k^B + \rho_{k+1}^A + \mathsf{val}_{k+1}^A]\\ &\geq 0 + 0 + 0\\ &= 1 - 3\alpha_k - 3\epsilon.\end{aligned}$$

k is odd and $\psi_{k-1}^B \geq n_0/2$. Then

$$\begin{aligned}&\mathbb{E}[\Psi_k^A - \Psi_{k-1}^A]\\ &= \mathbb{E}[\psi_k^A + \rho_{k+1}^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) + \mathsf{val}_{k+1}^A - \psi_{k-1}^A - \rho_k^A - \min(\psi_{k-1}^B + \rho_k^B, n_0/2) - \mathsf{val}_k^A]\\ &= \mathbb{E}[\rho_{k+1}^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \min(\psi_k^B, n_0/2) - \min(\psi_{k-1}^B + \rho_k^B, n_0/2)]\\ &\geq \mathbb{E}[\rho_{k+1}^A + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A + \min(\Lambda_k^B, 0)].\end{aligned}$$

Case 1: Message k is of the form $\mathsf{ECC}(z \in \Sigma^2, ?)$.

It holds that $z = \mathsf{C}(U_A)[k]$. Moreover, $\mathsf{val}_k^A = 0.5$ since $\psi_{k-1}^B \geq n_0/2$, so we want to show

$$\mathbb{E}[\rho_{k+1}^A + \mathsf{val}_{k+1}^A + \min(\Lambda_k^B, 0)] \geq 1 - 3\alpha_k - 3\epsilon.$$

Subcase 1.1: Bob interprets message k correctly.

If Bob entered Case 1, then $\mathsf{C}(U_A)[k] = \mathsf{C}(U_B)[k]$, which means $v(U_A[1:k]) = v(U_B[1:k])$ by Definition 7.8. If Bob entered Case 2 Subcase 2, then $v^* = v(U_A[1:k]) = v(U_B[1:k])$ In either case, since $t(v(U_B[1:k])) = \mathcal{T}$, Bob makes a neutral or positive update from his current complete correct transcript, so his next mes-

sage is always $\mathsf{ECC}(\mathsf{C}(v(U_B[1:k]), \bullet\bullet), ?)$ which has $\rho^A_{k+1} = 1$. Also, $\Lambda^B_k \geq 0$ by Lemma 7.11, so

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 1 + 0 + 0 \\ &\geq 1 - 3\alpha_k - 3\epsilon. \end{aligned}$$

If he entered Case 2 Subcase 3, he correctly decodes $v^* = v(U_A[1:k])$, and sends $\mathsf{ECC}(\mathsf{C}(U_A)[k], \delta \in \{0, 1, \leftarrow, ?\})$ with $\rho^A_{k+1} = 1$ with probability at least $1 - 6\alpha_k$ and otherwise $\rho^A_{k+1} + \mathsf{val}^A_{k+1} \geq 0.5$. Also, $\Lambda^B_k \geq 0$ by Lemma 7.11. This gives

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 1(1 - 6\alpha_k) + 0.5(6\alpha_k) + 0 \\ &\geq 1 - 3\alpha_k - 3\epsilon. \end{aligned}$$

Subcase 1.2: Bob interprets message k incorrectly.

If Bob enters Case 2 Subcase 3, he never updates, in which case $\Lambda^B_k = 0$. With probability at least $1 - p \geq 6(0.5 - \alpha_k - \epsilon)$, Bob sends $\mathsf{ECC}(z \in \Sigma^2, ?)$, so $\rho^A_{k+1} + \mathsf{val}^A_{k+1} \geq 0.5$. This gives

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 0.5 \cdot 6(0.5 - \alpha_k - \epsilon) + 0 \\ &= 1.5 - 3\alpha_k - 3\epsilon. \end{aligned}$$

Otherwise, his probability of updating is at most $3\alpha_k + 3\epsilon - 0.5$, so $\mathbb{E}[\Lambda^B_k] \geq 0.5 - 3\alpha_k - 3\epsilon$. Since he sends $\mathsf{ECC}(z \in \Sigma^2, ?)$, we have $\rho^A_{k+1} + \mathsf{val}^A_{k+1} \geq 0.5$ which gives

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 0.5 + 0.5 - 3\alpha_k - 3\epsilon \\ &= 1 - 3\alpha_k - 3\epsilon. \end{aligned}$$

Case 2: Message k is of the form $\mathsf{ECC}(z, \delta \in \{0, 1, \leftarrow\})$.

The message is not a question so $\mathsf{val}^A_k = 0$. Thus, we need to show

$$\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \geq 0.5 - 3\alpha_k - \epsilon.$$

Subcase 2.1: Bob interprets message k correctly.

He always sends a message $k+1$ of the form $\mathsf{ECC}(z, ?)$, so $\rho^A_{k+1} + \mathsf{val}^A_{k+1} \geq 0.5$. Then

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 0.5 - 0 \\ &\geq 0.5 - 3\alpha_k - \epsilon. \end{aligned}$$

Subcase 2.2: Bob interprets message k incorrectly.

Notice that $\alpha_k \geq \frac{1}{6}$ and so $\min(\Lambda^B_k, 0) > 0.5 - 3\alpha_k - 3\epsilon$. Then

$$\begin{aligned} &\mathbb{E}[\rho^A_{k+1} + \mathsf{val}^A_{k+1} + \min(\Lambda^B_k, 0)] \\ &\geq 0 - 0.5 - 3\alpha_k - \epsilon \\ &= 0.5 - 3\alpha_k - \epsilon. \end{aligned}$$

□

7.4.4 Concluding with Azuma's Inequality

Proof of Theorem 7.6. We defer the proof of communication complexity and computational complexity to Lemma 7.13. Here, we simply show that Protocol 7-1 is $\left(\frac{1}{6}, 1224\epsilon, 2 \cdot \exp\left(\frac{-\epsilon n_0}{800}\right)\right)$-scaling. First, the consistency property is clear: Alice never appends an operation to U_A such that the resulting transcript $t(v(U_A))$ is inconsistent with x. It suffices to show the two scaling properties. In particular, we will show that with probability at least $1 - \exp\left(-\frac{\epsilon n_0}{800}\right)$, both of the following statements hold for Alice:

- If $\alpha < \frac{1}{6} - 1224\epsilon$, then $t(v(U_A)) = \mathcal{T}$ and $w_A \geq \frac{K}{2}(1 - 6\alpha - 1224\epsilon)$.
- If $\alpha \geq \frac{1}{6} - 1224\epsilon$, then if $t(v(U_A)) \neq \mathcal{T}$ then $w_A \leq \frac{K}{2}(6\alpha - 1 + 1224\epsilon)$.

We call these the Alice-scaling conditions. By a similar analysis, the equivalent statements will hold for Bob as well. Then a union bound will give that the probability the scaling conditions hold simultaneously for both parties is at least $1 - 2 \cdot \exp(-\frac{\epsilon n_0}{800})$.

Let $\alpha_1, \ldots, \alpha_K$ denote the fractional number of corruptions in messages $1, \ldots, K$. Define

$$\mathcal{S}_k = \{i : i \leq k \wedge (i - 1 \in \mathcal{S} \vee i \in \mathcal{S} \vee i + 1 \in \mathcal{S})\}.$$

For $k \in \{1 \ldots K\}$, we define the random variables

$$\Phi^A_k = \Psi^A_k - 0.5k + 3k\epsilon + \sum_{i=1}^{k} 3\alpha_i + 10|\mathcal{S}_k|,$$

$$\Phi^B_k = \Psi^B_k - 0.5k + 3k\epsilon + \sum_{i=1}^{k} 3\alpha_i + 10|\mathcal{S}_k|.$$

By Lemma 7.12, for all k such that $k-1, k, k+1 \notin \mathcal{S}$,

$$\begin{aligned}\mathbb{E}[\Phi_k^A] &= \mathbb{E}\left[\Psi_k^A - 0.5k + 3k\epsilon + \sum_{i=1}^{k} 3\alpha_i + 10|\mathcal{S}_k|\right] \\ &\geq \mathbb{E}\left[\Psi_{k-1}^A - 0.5(k-1) + 3(k-1)\epsilon + \sum_{i=1}^{k-1} 3\alpha_i + 10|\mathcal{S}_k|\right] \\ &= \mathbb{E}[\Phi_{k-1}^A].\end{aligned}$$

For all k such that either $k-1 \in \mathcal{S}$, $k \in \mathcal{S}$, or $k+1 \in \mathcal{S}$,

$$\begin{aligned}\mathbb{E}[\Phi_k^A] &= \mathbb{E}\left[\Psi_k^A - 0.5k + 3k\epsilon + \sum_{i=1}^{k} 3\alpha_i + 10|\mathcal{S}_k|\right] \\ &\geq \mathbb{E}\left[\begin{array}{l}\Psi_{k-1}^A + \Lambda_k^A + \rho_k^A - \rho_{k-1}^A + \min(\psi_k^B + \rho_{k+1}^B, n_0/2) - \min(\psi_{k-1}^B + \rho_k^B, n_0/2) \\ + \mathsf{val}_{k+1}^A - \mathsf{val}_k^A - 0.5k + 3k\epsilon + \sum_{i=1}^{k-1} 3\alpha_i + 10|\mathcal{S}_{k-1}| + 10\end{array}\right] \\ &\geq \mathbb{E}[\Phi_{k-1}^A] - \left|\Lambda_k^A\right| - \left|\Lambda_k^B\right| - \left|\rho_k^B\right| - \left|\rho_{k-1}^B\right| - \left|\rho_k^A\right| - \left|\rho_{k-1}^A\right| - \left|\mathsf{val}_{k+1}^A\right| - \left|\mathsf{val}_k^A\right| - 0.5 + 3\epsilon + 3\alpha_k + 10 \\ &\geq \mathbb{E}[\Phi_{k-1}^A].\end{aligned}$$

Therefore, $\{\Phi_k^A\}_{k\geq 1}$ is a submartingale. A similar argument shows it has bounded distance

$$\begin{aligned}|\Phi_k^A - \Phi_{k-1}^A| &= \left|\Psi_k^A - \Psi_{k-1}^A - 0.5 + 3\epsilon + 3\alpha_k + |\mathcal{S}_k| - |\mathcal{S}_{k-1}|\right| \\ &\leq \left|\Lambda_k^A\right| + \left|\Lambda_k^B\right| + \left|\rho_k^B\right| + \left|\rho_{k-1}^B\right| + \left|\rho_k^A\right| + \left|\rho_{k-1}^A\right| + \left|\mathsf{val}_{k+1}^A\right| + \left|\mathsf{val}_k^A\right| + |-0.5 + 3\epsilon + 3\alpha_k| + 10 \\ &< 20.\end{aligned}$$

Similarly, Φ_k^B is a submartingale with bounded distance < 20. For convenience, define $\Phi_0^A = \Phi_0^B = -5$, and because $\Phi_1^A, \Phi_1^B \in [-1, 15]$, it still holds that Φ^A and Φ^B are submartingales. Moreover, recall that $|\mathcal{S}| \leq 20K\epsilon$ by Lemma 7.9 which implies that $|\mathcal{S}_K| \leq 60K\epsilon$.

We now show that the Alice-scaling conditions hold as long as $\Psi_K^A \geq R := n_0 + 2 + \frac{K}{2}(1 - 6\alpha - 1224\epsilon)$. Note that this implies that

$$\begin{aligned}\psi_K^A &= \Psi_K^A - \rho_{K+1}^A - \min(\psi_K^B + \rho_{K+1}^B, n_0/2) - \mathsf{val}_{K+1}^A \\ &\geq \Psi_K^A - n_0/2 - 2 \\ &\geq n_0/2 + \frac{K}{2}(1 - 6\alpha - 1224\epsilon).\end{aligned}$$

Then, by Lemma 7.10, if $\alpha < \frac{1}{6} - 1224\epsilon$, it holds that $\psi_K^A \geq n_0/2$ which means that Alice outputs $t(v(U_A)) = \mathcal{T}$ with weight $w_A \geq \frac{K}{2}(1 - 6\alpha - 1224\epsilon)$. On the other hand, if $\alpha \geq \frac{1}{6} - 1224\epsilon$, then either $t(v(U_A)) = \mathcal{T}$ or $\psi_K^A < n_0/2$, in which case $w_A \leq n_0/2 - \psi_K^A \leq \frac{K}{2}(6\alpha - 1 + 1224\epsilon)$.

Finally,

$$\begin{aligned}
\Pr\left[\Psi_K^A \geq R\right] &= 1 - \Pr\left[\Phi_K^A - \Phi_0^A < R - 0.5K + 3K\epsilon + \sum_{i=0}^{K} 3\alpha_i + 10|\mathcal{S}_K| - \Phi_0^A\right] \\
&\geq 1 - \Pr\left[\Phi_K^A - \Phi_0^A < R - 0.5K + 3K\epsilon + 3\alpha K + 600K\epsilon + 5\right] \\
&\geq 1 - \Pr\left[\Phi_K^A - \Phi_0^A < n_0 + 2 - \frac{K}{2}(1 - 6\alpha - 1224\epsilon) - 0.5K + 3K\epsilon + 3\alpha K + 600K\epsilon + 5\right] \\
&\geq 1 - \Pr\left[\Phi_K^A - \Phi_0^A < -n_0\right] \\
&\geq 1 - \exp\left(\frac{-\epsilon n_0}{800}\right).
\end{aligned}$$

The same calculation holds for Bob. It follows that Protocol 7-1 is $(\frac{1}{6}, 1224\epsilon, 2 \cdot \exp(-\frac{\epsilon n_0}{800}))$-scaling. □

7.4.5 Communication and Computational Complexity

Lemma 7.13. *The communication complexity of Protocol 7-1 is $O_\epsilon(n_0)$, and the computational complexity is $2^{2^{O_\epsilon(n_0)}}$.*

Proof. The communication complexity is $K \cdot M(|\Sigma|, \epsilon) = O_\epsilon(n_0)$.

As for the computational complexity, at the beginning, Alice and Bob agree on the code C. Each possible code is defined by a labeling of G; there are $4 \cdot (2^K - 1)$ edges with $|\Sigma|$ labels each, for $\leq |\Sigma|^{4 \cdot 2^K}$ possible codes. Both Alice and Bob choose the lexicographically first one that is an ϵ-sensitive layered code: ϵ-sensitivity can be checked in time $\text{poly}(|\Sigma|^K)$ by checking each word $w \in \Sigma^K$ and all possible prefix decodings. In each of the K rounds, the substantial actions that Alice (respectively Bob) performs are some subset of the following:

- Alice appends elements in $\{0, 1, \leftarrow, \bullet\}^2$ to U_A or appends elements in Σ^2 to P_A. These steps take time $\tilde{O}_\epsilon(1)$.
- Alice encodes $\mathsf{C}(U_A)$. This step takes time $\tilde{O}_\epsilon(n_0)$.
- Alice decodes $\mathsf{CDec}(P_A)$. She may need to test all 4^K possible paths, which could take time $\tilde{O}_\epsilon(n_0) \cdot 4^K$.
- Alice decodes a message m to the nearest $\mathsf{ECC}(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow, ?\}$ and computes the distance between m and $\mathsf{ECC}(z \in \Sigma^2, \delta \in \{0, 1, \leftarrow, ?\})$. Since $|\Sigma|$ and therefore the length of m is a constant independent of n_0, these steps take time $O_\epsilon(1)$.

In combination, the steps take total computational complexity $2^{2^{O_\epsilon(n_0)}}$ (where recall that $K = n_0/\epsilon$). □

Chapter 8

The Story for Erasures: Optimally Erasure Resilient Interactive Communication

In this chapter, 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 (see Theorem 2.3).

The main idea of our protocol is the following: In the setting of erasures, information can never be incorrect, only not received. Thus, a party should send their next bit of information repeatedly until the other party receives it. The tricky part is how to coordinate this while the two parties may only send one of two bits.

8.1 Formal Protocol

Protocol 8-1 : 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 $\beta = \perp$, or it is the first message in the block, 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 $\frac{1}{\epsilon} \in \mathbb{N}$.

8.2 Analysis

We begin by proving several claims about Protocol 8-1.

Claim 8.1. *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 8.2. *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 8.1, 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 8.3. *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 8.1, 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 8.4. *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 8.5. *Protocol 8-1 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 8.3 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.

□

Chapter 9

Future Directions

In this thesis, we have determined the optimal noise resilience for non-adaptive interactive coding schemes for both bit flip errors and erasures. In this final section, we leave with some questions to motivate future study.

9.1 Noise Resilient Interactive Coding Schemes

1. We have constructed protocols that are resilient to the optimal $\frac{1}{6}$ fraction of bit flip errors, or $\frac{1}{2}$ fraction of erasures. However, the strategies we employ in the two cases are very different. One may hope for a scheme that is resilient to *either* type of corruption. More precisely, we ask the following:

   (a) Is there an interactive coding scheme that is resilient to $\frac{1}{6}$ errors, *and* to $\frac{1}{2}$ erasures?
   (b) What is the best tradeoff (α, β, r) such that there exists a scheme resilient to any γ_{error} fraction of errors, and γ_{erasure} fraction of erasures, such that

   $$\alpha \cdot \gamma_{\text{error}} + \beta \cdot \gamma_{\text{erasure}} \leq r?$$

2. An interesting question is about the use of randomness in our bit-flip protocols in Chapters 3 and 7. In the large alphabet case, the optimal schemes are able to achieve optimal error resilience and positive rate *deterministically*; it is only when computational efficiency is desired that randomness is introduced [BK12, GH13]. By contrast, for the binary setting, our protocols require the use of randomness to obtain the optimal $\frac{1}{6}$ error resilience with *any* rate. Can one achieve $\frac{1}{6}$ error resilience deterministically?

9.2 Layered Codes

In this thesis, we have only defined and proven properties of layered codes that are useful in our construction of a positive rate interactive coding scheme. However, layered codes also serve as a generalization of tree codes that may be of independent interest, and we hope to see future work further generalizing the results of tree codes to this context. We propose a few problems to guide the future study of layered codes.

3. We have shown that *sensitive* layered codes exist, but have not addressed the analogue of tree codes. Do layered codes exist on any layered graph over Σ? Specifically, for any ϵ is there an assignment of the edges of a layered graph over Σ to a larger alphabet Σ_{out} such that for any two words $x, y \in \Sigma^n$ such that $v(x) \neq v(y)$, the suffix distance $\Delta_{sfx}(x, y) > 1 - \epsilon$?
4. Our protocol is one in which *layered* codes are necessary, and *tree* codes are not strong enough. Are there other contexts where this is the case? One possible use case may be in low memory settings, where a party cannot remember the full history of the messages they have sent, and so needing only to remember the vertex of the graph they are on may be useful.
5. Do tree codes beyond layered graphs? For example, does the definition of suffix distance generalize to any directed graph? Does Theorem 6.7 generalize to a more general context? Does Question 3 generalize?

Bibliography

[BE14] Mark Braverman and Klim Efremenko. List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 236–245, Los Alamitos, CA, USA, oct 2014. IEEE Computer Society.

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

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

[BK12] Zvika Brakerski and Yael Tauman Kalai. Efficient Interactive Coding against Adversarial Noise. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 160–166, 2012.

[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.

[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.

[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.

[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.

[Gel17] Ran Gelles. Coding for Interactive Communication: A Survey. *Foundations and Trends® in Theoretical Computer Science*, 13:1–161, 01 2017.

[GH13] Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 12 2013.

[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.

[GZ22] Meghal Gupta and Rachel Yun Zhang. The Optimal Error Resilience of Interactive Communication Over Binary Channels. In *Symposium on Theory of Computing, STOC 2012, New York, NY, USA, June 20 - June 24, 2022*, STOC '22. ACM, 2022.

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

[Sch92] Leonard 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.

[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.

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

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

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

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