Interactive Communication with Unknown Noise Rate[☆]

Varsha Dani[a,*], Mahnush Movahedi[a], Thomas P. Hayes[a], Jared Saia[a,1], Maxwell Young[b,2]

[a] *Department of Computer Science, University of New Mexico, Albuquerque, NM, USA*
[b] *Computer Science and Engineering Department, Mississippi State University, Starkville, MS, USA*

Abstract

Alice and Bob want to run a protocol over a noisy channel, where a certain number of bits are flipped adversarially. Several results take a protocol requiring L bits of noise-free communication and make it robust over such a channel. In a recent breakthrough result, Haeupler described an algorithm that sends a number of bits that is conjectured to be near optimal in such a model. However, his algorithm critically requires *a priori* knowledge of the number of bits that will be flipped by the adversary.

We describe an algorithm requiring no such knowledge. If an adversary flips T bits, our algorithm sends $L + O\left(\sqrt{L(T+1)\log L} + T\right)$ bits in expectation and succeeds with high probability in L. It does so without any *a priori* knowledge of T. Assuming a conjectured lower bound by Haeupler, our result is optimal up to logarithmic factors.

Our algorithm critically relies on the assumption of a private channel. We show that privacy is necessary when the amount of noise is unknown.

1. Introduction

How can two parties run a protocol over a noisy channel? Interactive communication seeks to solve this problem while minimizing the total number of bits sent. Recently, Haeupler [2] gave an algorithm for this problem that is conjectured to be optimal. However, as in previous work [3, 4, 5, 6, 7, 8, 9, 10], his algorithm critically relies on the assumption that the algorithm knows the noise rate in advance, *i.e.*, the algorithm knows in advance the number of bits that will be flipped by the adversary.

In this paper, we remove this assumption. To do so, we add a new assumption of privacy. In particular, in our model, an adversary can flip an unknown number of bits, at arbitrary times, but he never learns the value of any bits sent over the channel. This assumption is necessary: with a public channel and unknown

[☆] Some of the results of this paper appeared in the extended abstract [1] published in the Proceedings of the 42nd International Colloquium on Automata,Languages and Programming (ICALP 2015)

[*] Corresponding author

Email addresses: varsha@cs.unm.edu (Varsha Dani), movahedi@cs.unm.edu (Mahnush Movahedi), hayes@cs.unm.edu (Thomas P. Hayes), saia@cs.unm.edu (Jared Saia), myoung@cse.msstate.edu (Maxwell Young)

[1] This research was supported in part by NSF grant CNS-1318294.

[2] This research was supported in part by NSF grants CNS-1318294 and CCF-1420911.

noise rate, the adversary can run a man-in-the-middle attack to mislead either party (see Theorem 6.1, Section 6).

Problem Overview. We assume that Alice and Bob are connected by a noisy binary channel. Our goal is to build an algorithm that takes as input some distributed protocol π that works over a noise-free channel and outputs a distributed protocol π' that works over the noisy channel.

We assume an adversary chooses π, and which bits to flip in the noisy channel. The adversary knows our algorithm for transforming π to π'. However, he neither knows the private random bits of Alice and Bob, nor the bits sent over the channel, except when it is possible to infer these from knowledge of π and our algorithm.

We let T be the number of bits flipped by the adversary, and L be the length of π. As in previous work, we assume that Alice and Bob know L.

Our Results. Our main result is summarized in the following theorem.

Theorem 1.1. *Algorithm 3 tolerates an unknown number of adversarial errors, T, succeeds with high probability in the transcript length*[3], *L, and if successful, sends in expectation $L + O\left(\sqrt{L(T+1)\log L} + T\right)$ bits.*

The number of bits sent by our algorithm is within logarithmic factors of optimal, assuming a conjecture from [2] (see Theorem 6.3).

Results in this paper first appeared in conference proceedings [1].

1.1. Related Work

For L bits to be transmitted from Alice to Bob, Shannon [11] proposes an error correcting code of size $O(L)$ that yields correct communication over a *noisy* channel with probability $1 - e^{-\Omega(L)}$. At first glance, this may appear to solve our problem. But consider an *interactive* protocol with communication complexity L, where Alice sends one bit, then Bob sends back one bit, and so forth where the value of each bit sent *depends on the previous bits received.* Two problems arise. First, using block codewords is not efficient; to achieve a small error probability, "dummy" bits may be added to each bit prior to encoding, but this results in a superlinear blowup in overhead. Second, due to the interactivity, an error that occurs in the past can ruin all computation that comes after it. Thus, error correcting codes fall short when dealing with interactive protocols.

The seminal work of Schulman [12, 3] overcame these obstacles by describing a deterministic method for simulating interactive protocols on noisy channels with only a constant-factor increase in the total communication complexity. This work spurred vigorous interest in the area (see [13] for an excellent survey).

[3] Specifically with probability at least $1 - \frac{1}{L \log L}$

Schulman's scheme tolerates an adversarial noise rate of 1/240. It critically depends on the notion of a *tree code* for which an exponential-time construction was originally provided. This exponential construction time motivated work on more efficient constructions [7, 14, 15]. There were also efforts to create alternative codes [8, 16]. Recently, elegant computationally-efficient schemes that tolerate a constant adversarial noise rate have been demonstrated [5, 10]. Additionally, a large number of powerful results have improved the tolerable adversarial noise rate [4, 6, 9, 17, 18].

The closest prior work to ours is that of Haeupler [2]. His work assumes a fixed and known adversarial noise rate ϵ, the fraction of bits flipped by the adversary. Communication efficiency is measured by *communication rate* which is L divided by the total number of bits sent. Haeupler [2] describes an algorithm that achieves a communication rate of $1 - O(\sqrt{\epsilon \log \log(1/\epsilon)}$, which he conjectures to be optimal. We compare our work to his in Section 6.

Feinerman, Haeupler and Korman [19] recently studied the interesting related problem of spreading a single-bit rumor in a noisy network. In their framework, in each synchronous round, each agent can deliver a single bit to a random anonymous agent. This bit is flipped independently at random with probability $1/2 - \epsilon$ for some fixed $\epsilon > 0$. Their algorithm ensures with high probability that in $O(\log n/\epsilon^2)$ rounds and with $O(n \log n/\epsilon^2))$ messages, all nodes learn the correct rumor. They also present a majority-consensus algorithm with the same resource costs, and prove these resource costs are optimal for both problems.

1.2. Formal Model

Our algorithm takes as input a protocol π which is a sequence of L bits, each of which is transmitted either from Alice to Bob or from Bob to Alice. As in previous work, we also assume that Alice and Bob both know L. We let Alice be the party who sends the first bit in π.

Channel Steps. We assume communication over the channel is synchronous and individual computation is instantaneous. We define a *channel step* as the amount of time that it takes to send one bit over the channel.

Silence on the Channel. When neither Alice nor Bob sends in a channel step, we say that the channel is silent. In any contiguous sequence of silent channel steps, the bit received on the channel in the first step is set by the adversary for free. By default, the bit received in subsequent steps of the sequence remains the same, unless the adversary pays for one bit flip in order to change it. In short, the adversary pays a cost of one bit flip each time it wants to change the value of the bit received in any contiguous sequence of silent steps.

1.3. Overview of Our Result

Challenges. Can we adapt prior results by guessing the noise rate? Underestimation threatens correctness if the actual number of bit flips exceeds the algorithm's tolerance. Conversely, overestimation leads to sending more bits than necessary. Thus, we need a protocol that adapts to the adversary's actions.

One idea is to adapt the amount of communication redundancy based on the number of errors detected thus far. However, this presents a new challenge because the parties may have different views of the number of errors. They will need to synchronize their adaptions over the noisy channel. This is a key technical challenge to achieving our result.

Another technical challenge is termination. The length of the simulated protocol is necessarily unknown, so the parties will likely not terminate at the same time. After one party has terminated, it is a challenge for the other party to detect this fact based on bits received over the noisy channel.

A high-level overview of how we address these challenges is given in Section 2.4.

1.4. Paper Organization

The rest of this paper is organized as follows. In Section 2, we describe a simple algorithm for interactive communication that works when $T = O(L/\log L)$. We analyze this algorithm in Section 3. In Section 4, we describe an algorithm for interactive communication that works for any finite T; we prove this algorithm correction in Section 5. Section 6 gives some relevant remarks, including justifying private channels and comparing our algorithm with past work. Finally, we conclude and give directions for future work in Section 7.

2. Bounded T - Algorithm

In this section, we describe an algorithm that enables interactive communication problem when $T = O(L/\log L)$.

2.1. Overview, Notation and Definitions

Our algorithm is presented as Algorithm 1. The overall idea of the algorithm is simple: the parties run the original protocol π for a certain number of steps as if there was no noise. Then, Alice determines whether an error has occurred by checking a fingerprint from Bob. Based on the result of this verification, the computation of π either moves forward or is rewound to be performed again.

2.2. Helper Functions

Before giving details of the algorithm, we first describe some helper functions and notation (see Figure 1).

Fingerprinting. To verify communication, we make use of the following well-known theorem.

Theorem 2.1. *[Naor and Naor [20]] For any positive integer $\mathcal{L}$ and any probability p, there exists a hash function $\mathcal{F}$ that given a uniformly random bit string S as the seed, maps any string of length at most $\mathcal{L}$ bits to a bit string hash value H, such that the collision probability of any two strings is at most p, and the length of S and H are $|S| = \Theta(\log(\mathcal{L}/p))$ and $|H| = \Theta(\log(1/p))$ bits.*

L	The length of the protocol to be simulated.
π	The L-bit protocol to be simulated, augmented by random bits to length $\left(1 + \left\lceil \frac{L}{R_0} \right\rceil\right) R_0$.
$\pi[\mathcal{T}, \ell]$	The result of the computation of the next ℓ bits of π after history $\mathcal{T}$.
R_0	Initial round size in the algorithm. This is the smallest power of 2 that is greater than $\sqrt{LF}$. So $\sqrt{LF} \leq R_0 \leq 2\sqrt{LF}$
F	The length of the fingerprint.
$\mathcal{T}_a$	Alice's tentative transcript.
$\mathcal{T}_b$	Bob's tentative transcript.
$\mathcal{T}_a^*$	Alice's verified transcript.
$\mathcal{T}_b^*$	Bob's verified transcript.
$\mathcal{T}[0 : \ell]$	The first ℓ bits of $\mathcal{T}$. If $\lvert\mathcal{T}\rvert < L$ this is *null*

Figure 1: Glossary of Notation

We define two functions based on this theorem, h and MatchesFP. In this section, we will write h_L to denote that the probability of error p is polynomial in L. In particular, we can set $p = 1/L^2$, with fingerprints of size $O(\log L)$. The function $\mathrm{h}_L(T)$ takes a transcript T and returns a tuple (s, f), where s is uniformly random bit string and f is the output of the hash function $\mathcal{F}$ in the theorem above when given inputs s and T. We refer to this tuple as the *fingerprint* of T.

The function $\mathrm{MatchesFP}((s, f), T)$ takes a fingerprint (s, f) and a transcript T. It returns true if and only if the output of $\mathcal{F}$ when given bit string s and transcript T is equal to the value f. In both of these functions, the total length of the fingerprint is given by the value F, which will be defined later.

Algebraic Manipulation Detection Codes. Our result makes critical use of Algebraic Manipulation Detection (AMD) Codes from [21]. These codes provide three functions: amdEnc, IsCodeword and amdDec. The function $\mathrm{amdEnc}(m)$ creates an encoding of a message m. The function $\mathrm{IsCodeword}(m')$ returns true if and only if a received message m' is equal to $\mathrm{amdEnc}(m)$ for some sent message m. The function $\mathrm{amdDec}(m')$ takes a received value m', where $\mathrm{IsCodeword}(m')$, and returns the value m such that $\mathrm{amdEnc}(m) = m'$. Intuitively, AMD Codes enable detection of bit corruptions on encoded words, with high probability.

We make use of the following theorem about AMD codes. This is a slight rewording of a theorem from [21].

Theorem 2.2. *[21] For any $\delta > 0$, there exists functions* amdEnc, IsCodeword *and* amdDec*, such that, for any bit string m of length x:*

- $\mathrm{amdEnc}(m)$ *is a string of length $x + C\log(1/\delta)$, for some constant C;*

- IsCodeword(amdEnc(m)) *and* amdDec(amdEnc(m)) $= m$*;*

- *For any bit string* $s \neq 0$ *of length* x*,* $Pr(\text{IsCodeword}(\text{amdEnc}(m) \oplus s)) \leq \delta$

In this section, we set $\delta = 1/L^2$ and add $O(\log L)$ additional bits to the message word. Also in this section, we will always encode strings of size $O(\log L)$, so the AMD encoded messages will be of size $O(\log L)$.

In the algorithm, we will denote the fixed length of the AMD-encoded fingerprint by F.

2.3. Remaining Notation

Transcripts. We define Alice's *tentative transcript*, $\mathcal{T}_{\mathcal{A}}$, as the sequence of possible bits of π that Alice has either sent or received up to the current time. Similarly, we let $\mathcal{T}_{\mathcal{B}}$ denote Bob's transcript. For both Alice or Bob, we define a *verified transcript* to be the longest prefix of a transcript for which a verified fingerprint has been received. We denote the verified transcript for Alice as $\mathcal{T}^*_{\mathcal{A}}$, and for Bob as $\mathcal{T}^*_{\mathcal{B}}$. The notation $T \preccurlyeq T'$ signifies that a transcript T is a prefix of a transcript T'.

Rounds. We define one of *Alice's rounds* as one iteration of the repeat loop in Alice's protocol. Alice's round consists of r_a channel steps, where r_a is the *round size* value maintained by Alice. Similarly, we define one of *Bob's rounds* as one iteration of the repeat look in Bob's protocol. Such a round consists of r_b channel steps, where r_b is the *round size* for Bob.

Other Notation. For a transcript $\mathcal{T}$ and integer i, we define $\mathcal{T}[0 : i]$ to be the first i bits of $\mathcal{T}$. For two strings x and y, we define $x \odot y$ to be the concatenation of x and y.

2.4. Algorithm Overview

To facilitate discussion of the algorithm, we first state some important properties of rounds (proven in Section 3). First, the size of any round is always a power of two. Second, the start of each of Bob's rounds always coincides with the start of one of Alice's rounds. This ensures that whenever Bob is listening for the message $\mathcal{F}'_a$, Alice will be sending such a message.

We first describe one of Alice's rounds in which 1) neither Alice nor Bob terminate; and 2) there are no adversarial bit flips. In such a round, Alice sends an encoded message containing two pieces of information. These are m_a, which is the number of failed rounds Alice has counted so far; and $|\mathcal{T}^*_a|$, which is the size of Alice's verified transcript.

When Bob decodes this message, he synchronizes several values with Alice. In particular, he sets his round size value, r_b, and mistake estimate value, m_b, so they equal the values Alice sent. Then, based on $|\mathcal{T}^*_a|$, Bob either increases the length of his verified transcript, or else decreases the length of his tentative transcript. After this synchronization, Alice and Bob both compute a certain number of bits of π and add

Algorithm 1: Bounded Error Interactive Communication

ALICE'S PROTOCOL

```
$\mathcal{T}_a \leftarrow null$; $\mathcal{T}_a^* \leftarrow null$;
$m_a \leftarrow 0$; $r_a \leftarrow R_0$;
repeat
    $\mathcal{F}_a \leftarrow \text{amdEnc}(m_a, r_a, |\mathcal{T}_a^*|)$;
    Send $\mathcal{F}_a$;
    Append $\pi[\mathcal{T}_a, r_a - 2F]$ to $\mathcal{T}_a$;
    Receive Bob's $F$-bit message, $\mathcal{F}_b'$;
    if IsCodeword($\mathcal{F}_b'$) then
        if $|\mathcal{T}_a^*| \geq L$ then
            Output $\mathcal{T}_a^*[0 : L]$ and
            Terminate;
        $\mathcal{F} \leftarrow \text{amdDec}(\mathcal{F}_b')$;
        if MatchesFP($\mathcal{F}, \mathcal{T}_a$) then
            // successful round;
            $\mathcal{T}_a^* \leftarrow \mathcal{T}_a$;
    else
        // round failed;
        $\mathcal{T}_a \leftarrow \mathcal{T}_a^*$;
        $m_a \leftarrow m_a + 1$;
        if $1 + m_a$ is a power of 4 then
            $r_a \leftarrow r_a/2$;
until $m_a = \frac{R_0^2}{4F^2} - 1$;
```

BOB'S PROTOCOL

```
$\mathcal{T}_b \leftarrow null$; $\mathcal{T}_b^* \leftarrow null$;
$m_b \leftarrow 0$; $r_b \leftarrow R_0$;
repeat
    Receive Alice's $F$-bit message, $\mathcal{F}_a'$;
    if all bits of $\mathcal{F}_a'$ are equal then
        // Alice has likely left;
        Output $\mathcal{T}_b^*[0 : L]$ and
        Terminate;
    if IsCodeword($\mathcal{F}_a'$) then
        $(m, r, \ell) \leftarrow \text{amdDec}(\mathcal{F}_a')$;
        // synchronize values;
        $r_b \leftarrow r$;
        $m_b \leftarrow m$;
        if $\ell > |\mathcal{T}_b^*|$ then
            $\mathcal{T}_b^* \leftarrow \mathcal{T}_b$;
        else
            $\mathcal{T}_b \leftarrow \mathcal{T}_b^*$;
        Append $\pi[\mathcal{T}_b, r_b - 2F]$ to $\mathcal{T}_b$;
        $\mathcal{F}_b \leftarrow \text{amdEnc}(\text{h}_L(\mathcal{T}_b))$;
        Send $\mathcal{F}_b$;
    else
        // corruption occurred;
        Send random bits for $r_b - F$ steps;
        $m_b \leftarrow m_b + 1$ ;
        if $1 + m_b$ is a power of 4 then
            $r_b \leftarrow r_b/2$;
until $m_b = \frac{R_0^2}{4F^2} - 1$;
```

these to their tentative transcripts. Finally Bob sends an encoded fingerprint to Alice. She verifies this fingerprint, and then adds the bits of π computed during this round to her verified transcript.

There are two key ways in which adversarial bit flips can alter the above scenario. First, when the encoded message Alice sends containing m_a and $|\mathcal{T}_a^*|$ is corrupted. In this case, Bob will send random bits for the remainder of the round. This ensures two things. First, whenever Alice is listening for a fingerprint from Bob, Bob will either be sending a fingerprint or random bits. Thus, with high probability, the adversary will be unable to forge an encoding of a fake fingerprint by flipping bits. Second, Bob's error count updates at the same time as Alice's.

The other key way in which adversarial bit flips can alter the ideal scenario is as follows. The adversary flips bits in such a way that the encoded fingerprint, $\mathcal{F}_b'$ that Bob sends to Alice, fails to be a valid fingerprint for Alice's tentative transcript. In this case, Alice rewinds her tentative transcript, increments her error count, and updates her block size.

Handling Termination. In previous work, since ϵ and L' are known, both parties know when to terminate (or *leave* the protocol), and can do so at the same time. However, since we know neither parameter, termination is now more challenging.

In our algorithm, π is augmented with a certain number of additional bits that Alice sends to Bob. Each of these bits is set independently and uniformly at random by Alice. Alice terminates when her verified transcript is of length greater than L. Bob terminates when he receives a value $\mathcal{F}_a'$, where all bits are the same. This conditions ensures that 1) Bob is very unlikely to terminate before Alice; and 2) Bob terminates soon after Alice, unless the adversary pays a significant cost to delay this.

3. Bounded T - Analysis

We now prove that with high probability, Algorithm 1 correctly simulates π when T is promised to be $O(L/\log L)$. Before proceeding to our proof, we define two bad events.

> *Hash Collision.* Either Alice or Bob incorrectly validates a fingerprint and updates their verified transcript to include bits not in π.
>
> *Failure of AMD Codes* The adversary corrupts an encoded message into the encoding or a different message. Or the encoding of some message, after possible adversary corruption, equals a bit string of all zeroes or all ones.

Throughout this section, we will assume neither event occurs. At the end of this section, we will show that the probability that either even occurs is polynomially small in L.

Lemma 3.1. *Each player's round size is always a power of two.*

Proof. This is immediate from the fact that the round size starts out as a power of 2 and the fact that each time it decreases, it decreases by a factor of 2. □

Lemma 3.2. *m_a is monotonically increasing, and hence Alice's round size never increases.*

Proof. This follows immediately from the fact that the only time m_a changes is on Line 15 of Alice's protocol, when it is incremented by 1. □

Lemma 3.3. *Algorithm 1 has the following properties:*

1. *When Bob starts a round, Alice starts a round,*
2. *$m_b \leq m_a$ at all times that Alice remains in the protocol.*

Proof. This follows by induction on m_a.

Base Case. We first show that the lemma holds while $m_a = 0$. Note that m_b can only increase after Bob has spent a round sending random bits. During such a round, Alice will increment m_a before Bob increments m_b. Next, note that while $m_b = m_a = 0$, Alice and Bob both have the same round sizes, and so when Bob starts a round, Alice starts a round.

Inductive Step. Consider the channel step, t, at which Alice increases m_a to some value $j > 0$. We must show that the lemma statement holds throughout the time while $m_a = j$. By the inductive hypothesis, up to time t, $m_b \leq m_a$, and when Bob started a round, Alice started a round. There are two cases for the value of m_b at the end of channel step t.

Case 1. $m_b < j$. In this case, Bob must not have received $\mathcal{F}_a$ at the beginning of the round he is in at channel step t. Hence, Bob transmits random bits during this entire round. Bob's round size is an integer multiple of Alice's round size (by Lemma 3.1). Thus, Bob will transmit random bits throughout Alice's round begun at channel step $t + 1$. So Alice will not receive a matching fingerprint at the end of the round she began at step $t + 1$, and so she will increment m_a before Bob increments m_b. This will happen before Bob completes the round he is in at time t, so both conditions of the lemma hold while $m_a = j$.

Case 2. $m_b = j$. Note that m_b can only increase after Bob has spent a round sending random bits. During such a round, Alice will increment m_a before Bob increments m_b. Thus, while $m_a = j$, $m_b = j$. Next, note that, if $m_b = m_a = j$ at step t, then Alice and Bob both ended their rounds at step t. Hence, during the time that $m_a = j$, when Bob starts a round, Alice starts a round. □

The following corollaries are immediate from the above lemma.

Corollary 3.4. *When Bob ends a round, Alice ends a round.*

Corollary 3.5. *Bob's rounds are at least as large as Alice's rounds.*

The following corollary holds from the above lemma and the fact that Bob's round sizes are at least as large as Alice's.

Corollary 3.6. *While both parties remain in the protocol, whenever Bob is listening for a $\mathcal{F}_a$, Alice is sending it. Also, whenever Alice is listening for $\mathcal{F}_b$, either Bob is sending it, or Bob is sending random bits.*

The following lemma also follows from Lemma 3.3.

Lemma 3.7. *Let $\mathcal{R}$ be one of Alice's rounds which starts and ends at the same time as one of Bob's rounds. Then, at the end of $\mathcal{R}$, either $m_a - m_b$ is the same as it was at the beginning of $\mathcal{R}$ or it equals* 0 *or* 1.

Proof. If $\mathcal{F}_a$ is corrupted at the beginning of $\mathcal{R}$, Bob transmits random bits for the rest of $\mathcal{R}$, and both Alice and Bob increment their error counts at the end, so $m_a - m_b$ stays the same.

If $\mathcal{F}_a$ is not corrupted at the beginning of $\mathcal{R}$, then Bob sets m_b to m_a at the beginning of $\mathcal{R}$, so at the end, $m_a - m_b \leq 1$. By Lemma 3.3 (2), $m_a - m_b \geq 0$. □

3.1. Phases

We now give some definitions.

Definition 3.8. We define *phase* j to be all of Alice's rounds of size $R_0/2^j$.

Definition 3.9. We define Δ_j, for all $j > 0$, to be the value $m_a - m_b$ at the end of phase j.

Note that at the beginning of phase j, Alice's error count is $4^j - 1$. We now give a few lemmas about phases.

Lemma 3.10. *For any $j > 0$, phase j contains at least $3\Delta_{j-1}$ of Alice's rounds,*

Proof. Consider any $j > 0$. At the beginning of phase j, $m_a = 4^j - 1$. Also, at the beginning of phase j, by Lemma 3.3 (2), $m_b \leq m_a$. Hence, $0 \leq \Delta_{j-1} \leq 4^j - 1$. Note that m_a increases by at most 1 in each of Alice's rounds. Thus, $3\Delta_{j-1}$ rounds after the beginning of phase j, the value of m_a is at most:

$$\begin{aligned} 4^j - 1 + 3\Delta_{j-1} &\leq 4^j - 1 + 3(4^j - 1) \\ &< 4^{j+1} - 1 \end{aligned}$$

Thus after $3\Delta_{j-1}$ rounds, m_a is not large enough for Alice to advance to phase $j+1$. □

Progressive, Corrupted and Wasted Rounds. Let $\mathcal{R}$ be one of Alice's rounds. We call $\mathcal{R}$ *progressive* if Alice does not update her error count during the round, or equivalently if her verified transcript length increases. We call $\mathcal{R}$ *corrupted* if the adversary flipped at least one bit in the round. We call $\mathcal{R}$ *wasted* if it is neither progressive nor corrupted. We want to bound the number of wasted rounds since this number represents amount by which m_a is potentially an overestimate of T.

We note that wasted rounds occur only when $r_b > r_a$. In this case, Bob is not listening when Alice sends him $\mathcal{F}_a$. As a result, Bob does not send Alice a valid fingerprint at the end of her round, and so her verified transcript does not increase, even though the adversary has not flipped any bits.

The following lemma bounds the number of wasted rounds in a phase, and gives other critical properties.

Lemma 3.11. *Suppose at the beginning of phase j, $j > 0$, Bob is at the start of a round and his round size is at most $R_0/2^{j-1}$. Then*

1. *There are at most Δ_{j-1} wasted rounds in phase j;*
2. $\Delta_j \in \{0, 1, 2\Delta_{j-1}\}$*; and*
3. *Bob ends a round at the end of phase j.*

Proof. If Bob's round size initially less than $R_0/2^{j-1}$, then it must equal $R_0/2^j$ in order to be a power of two. Hence Alice and Bob will have rounds that are the same size for the entire phase, and the lemma holds trivially.

We now consider the harder case where Bob's round size equals $R_0/2^{j-1}$.

By Definition 3.8, Alice has round size $R_0/2^j$ throughout phase j. By Lemma 3.3 (2), Bob's round size is always greater than or equal to Alice's round size. Thus, as soon as 1) Bob receives $\mathcal{F}_a$ in one of his rounds in phase j, or 2) Bob sets m_b equal to Alice's error count at the beginning of phase j, then Bob's round size will be $R_0/2^j$ for the remainder of the phase. Finally, by Lemma 3.3 (1), from that point on, Alice and Bob will begin, and thus end, all rounds at the same time.

Now consider Bob's rounds in phase j. Assume the adversary corrupts $\mathcal{F}_a$ in Bob's rounds 1 through i for some value $i \geq 0$, and then the adversary does not corrupt $\mathcal{F}_a$ in Bob's round $i+1$. We consider two cases.

Case 1: $i < \Delta_{j-1}$. Each of the first i rounds of Bob spans two rounds of Alice. By Lemma 3.10, these rounds are all contained in phase j. Consider each pair of Alice's rounds spanned by one of Bob's rounds. The first round in the pair is corrupted, but during the second, Bob is transmitting random bits and Alice will not receive a fingerprint from him. Thus, this round is wasted. Hence, there are i wasted rounds.

In round $i+1$, Bob synchronizes his round size with Alice since he receives $\mathcal{F}_a$. Thus, there are no more wasted rounds. Applying Lemma 3.7 for the remaining rounds of the phase, we see that at the end of the phase, $m_a - m_b = \Delta_j$ is either 0 or 1.

Case 2: $i \geq \Delta_{j-1}$. Bob increases m_b by 1 in each of his first i rounds. Note that at the beginning of phase j, Alice's error count is $4^j - 1$. Thus, after Bob's first i rounds, $m_b = (4^j - 1) - \Delta_{j-1} + i$. Hence when $i = \Delta_{j-1}$, $m_b = (4^j - 1)$. At that time, Bob sets his round size to $R_0/2^j$, and so Alice and Bob will have the same round sizes, and will hence begin and end all rounds at the same step, for the rest of phase j. Thus, there are no more wasted rounds. Note that in this case, at Bob's Δ_{j-1} round, $m_a - m_b$ will be $2\Delta_{j-1}$. Applying Lemma 3.7 for the remaining rounds of the phase, we see that $\Delta_j = 2\Delta_{j-1}$, or Δ_j is 0 or 1. □

Lemma 3.12. *For every $j \geq 0$:*

1. *There are at most 2^{j-1} wasted rounds in phase j;*
2. *$\Delta_j \leq 2^j$; and*
3. *Bob ends a round at the end of phase j.*

Proof. We prove this by induction on j.

Base Case. At the beginning of phase 0, Bob is at the start of a round and his round size is R_0. Thus, by Lemma 3.11: there are 0 wasted rounds in phase 0; $\Delta_0 \leq 1$; and Bob ends a round at the end of phase 0.

Inductive Step. Consider some $j > 0$. By the inductive hypothesis, $\Delta_{j-1} \leq 2^{j-1}$. At the beginning of phase j, $m_b = m_a - \Delta_{j-1} \leq (4^j - 1) - \Delta_{j-1}$, so that $r_b = R_0/2^{\lfloor \log_4 (1+m_b) \rfloor} \leq R_0/2^{\lfloor \log_4 (4^j - \Delta_{j-1}) \rfloor} \leq R_0/2^{j-1}$. The last line holds since $0 \leq \Delta_{j-1} \leq 2^{j-1}$.

Also, by the inductive hypothesis, Bob ended a round at the end of phase $j-1$, and so is starting a round at the beginning of phase j. Hence, we can apply Lemma 3.11 to phase j. From this lemma, it follows that 1) the number of wasted rounds in phase j is at most 2^{j-1}; 2) $\Delta_j \leq 2\Delta_{j-1} \leq 2^j$; and 3) Bob ends a round at the end of phase j. □

Note from the above lemma that Bob's rounds are never more than double the size of Alice's rounds. The following lemma sums up what we now know about Alice and Bob's rounds.

Lemma 3.13. *The following are always true.*

1. *Bob's round size is either equal to Alice's round size or double Alice's round size.*
2. *If Bob's round size equals Alice's round size, then when Alice starts a round, Bob starts a round.*
3. *If Bob's round size is twice Alice's round size, then when Alice starts a round, either Bob starts a round, or Bob is in the middle of a round.*

Proof. The lemma follows from Corollary 3.5, Lemma 3.3, and Lemma 3.12. □

Lemma 3.14. *It is always the case that $\mathcal{T}_a^* \preccurlyeq \pi$, where π is the padded transcript.*

Proof. This holds by Lemma 3.25 and Lemma 3.26 and the fact that Alice never adds any string to $\mathcal{T}_a^*$ that is not verified by an encoded fingerprint from Bob. □

Lemma 3.15. *At the beginning and end of each of Alice's rounds,*

$$\mathcal{T}_b^* \preccurlyeq \mathcal{T}_a^* = \mathcal{T}_a \preccurlyeq \mathcal{T}_b;$$

where at most one of the inequalities is strict. Moreover, at the end of a channel step in which Bob receives $\mathcal{F}_a$ correctly,

$$\mathcal{T}_b^* = \mathcal{T}_b = \mathcal{T}_a^*.$$

Proof. We prove this by induction on Alice's round number.

Base Case. At the beginning of the algorithm, all transcripts are *null*, so $\mathcal{T}_b^* = \mathcal{T}_a^* = \mathcal{T}_a = \mathcal{T}_b$. Moreover if Bob receives $\mathcal{F}_a$ correctly in this round, then $\mathcal{T}_b^* = \mathcal{T}_b = \mathcal{T}_a^*$.

Inductive Step. We must show that the lemma holds for the j-th round. By the inductive hypothesis, at the end of the $j-1$-th round,

$$\mathcal{T}_b^* \preccurlyeq \mathcal{T}_a^* = \mathcal{T}_a \preccurlyeq \mathcal{T}_b,$$

with at most one of the inequalities being strict. Clearly the statement about the inequalities will thus hold at the beginning of the j-th round.

Alice's j-th round starts with Alice sending Bob $\mathcal{F}_a$.

Case 1: Bob does not receive $\mathcal{F}_a$. If Bob does not receive $\mathcal{F}_a$, then either 1) he was listening and it was corrupted; or 2) he was not listening for it. If he was listening and $\mathcal{F}_a$ was corrupted, then Bob transmits random bits for the remainder of his round, which will be the remainder of Alice's round by Lemma 3.13. By the same lemma, if Bob was not listening, then he must be in the middle of a round that is twice as large as Alice's. In either case, Bob transmits random bits for the remainder of Alice's j-th round.

Thus, Alice does not receive a matching fingerprint from Bob at the end of her j-th round. Thus, at the end of her round, $\mathcal{T}_a \leftarrow \mathcal{T}_a^*$ and $\mathcal{T}_b$ and $\mathcal{T}_b^*$ are unchanged. Hence, it continues to hold that:

$$\mathcal{T}_b^* \preccurlyeq \mathcal{T}_a^* = \mathcal{T}_a \preccurlyeq \mathcal{T}_b;$$

and at most one of the inequalities is strict.

Case 2: Bob receives $\mathcal{F}_a$. If Bob receives $\mathcal{F}_a$, then he learns the length of $\mathcal{T}_a^*$ and also Alice's round size. By the inductive hypothesis, either $\mathcal{T}_a^* = \mathcal{T}_b^*$ or $\mathcal{T}_a^* = \mathcal{T}_b$. Based on the length of $\mathcal{T}_a^*$, Bob either updates $\mathcal{T}_b^*$ or rewinds $\mathcal{T}_b$, so that $\mathcal{T}_b^* = \mathcal{T}_b = \mathcal{T}_a^*$. This establishes the second part of the lemma for the j-th round.

Next Alice and Bob continue their rounds which are the same size. If Alice receives a correct fingerprint from Bob at the end of her round, then the following holds:

$$\mathcal{T}_b^* \preccurlyeq \mathcal{T}_a^* = \mathcal{T}_a = \mathcal{T}_b.$$

If Alice does not receive a correct fingerprint from Bob at the end of her round, then the following holds:

$$\mathcal{T}_b^* = \mathcal{T}_a^* = \mathcal{T}_a \preccurlyeq \mathcal{T}_b.$$

In either case, the first part of the lemma statement holds at the end of Alice's j-th round. □

Lemma 3.16. *Bob leaves after Alice. When Alice leaves, $|\mathcal{T}_b^*| \geq L$.*

Proof. Bob leaves only when he receives an $\mathcal{F}_a'$ that is all zeroes or all ones. By Lemma 3.26, $\mathcal{F}_a'$ is never such a string, and the adversary cannot convert $\mathcal{F}_a$ to such a string by bit flipping. It follows that Bob receives such a string only after Alice has left.

Alice leaves only when 1) she has received an encoded fingerprint from Bob; and 2) $|\mathcal{T}_a^*| \geq L$. If Alice receives a correctly encoded fingerprint from Bob, then by Lemma 3.26, Bob must have sent one, and hence Bob must be in a round where he received $\mathcal{F}_a$ correctly. By Lemma 3.15, at that channel step, $\mathcal{T}_b^* = \mathcal{T}_b = \mathcal{T}_a^*$. Hence at the step when Alice receives the encoded fingerprint from Bob, $\mathcal{T}_b^* = \mathcal{T}_a^*$. Thus, when Alice leaves, $|\mathcal{T}_b^*| \geq L$. □

Lemma 3.17. *When either party terminates, their output is correct.*

Proof. The proof follows from Lemmas 3.14, 3.15, and 3.16, and the fact that when either party terminates, they output the first L bits of their verified transcript. □

3.3. Cost

Lemma 3.18. *After Alice leaves, the adversary must flip at least one bit for each of Bob's rounds that does not result in Bob leaving.*

Proof. After Alice has left, there is silence on the channel in the steps when Bob is listening for Alice's encoded message. This means that if there is no bit flipping by the adversary, the channel transmits the same bit in every channel step, causing Bob to read a string of all zeroes or all ones, and terminate. Thus, the adversary must flip at least one bit each time Bob is listening for a codeword. □

Lemma 3.19. *There are at most $2^j - 1$ wasted rounds prior to the end of phase j, for all $j \geq 0$.*

Proof. This follows trivially by repeated applications of Lemma 3.12 (1). □

Throughout this section, we assume the worst case, that the adversary corrupts at most one bit per corrupted round.

Lemma 3.20. *At all times, $m_a \leq T + \sqrt{T}$. In particular, there are no more than $\sqrt{T}$ wasted rounds.*

Proof. By way of contradiction, assume $m_a > T + \sqrt{T}$ at some step, in some phase j, $j \geq 0$. Then the number of wasted rounds at this step must be greater than $\sqrt{T}$. But by Lemma 3.19, the number of wasted rounds at the end of phase j is no more than $2^j - 1$. Thus, we have $\sqrt{T} < 2^j - 1$, or $T < (2^j - 1)^2$.

But m_a is no larger than the number of corrupted rounds plus the number of wasted rounds. By the above paragraph, $T < (2^j - 1)^2$ and the number of wasted rounds is no more than $2^j - 1$. Thus $m_a < (2^j - 1)^2 + (2^j - 1)$. Moreover, we know that in phase j, $m_a \geq 4^j - 1$. Thus, we know

$$4^j - 1 < (2^j - 1)^2 + (2^j - 1).$$

Simplifying, we get $2^j < 1$, which is a contradiction for any $j \geq 0$. □

Let m_a^* denote Alice's error count when she leaves the algorithm, and m_b^* denote Bob's error count when he himself leaves the algorithm.

Lemma 3.21. *Alice terminates in at most $L + O(\sqrt{LF(1 + m_a^*)})$ steps.*

Proof. We first calculate the cost of the rounds that are not progressive for Alice. The number of non-progressive rounds that she has executed is m_a^*. Her cost for these rounds is at most the following.

$$\begin{aligned}
\sum_{i=1}^{m_a^*} \frac{R_0}{2^{\lfloor \log_4 i \rfloor}} &\leq 2R_0 \sum_{i=1}^{m_a^*} \frac{1}{2^{\log_4 i}} \\
&= 2R_0 \sum_{i=1}^{m_a^*} \frac{1}{\sqrt{i}} \\
&\leq 2R_0 \int_0^{m_a^*} \frac{1}{\sqrt{i}} \\
&= 4R_0 \sqrt{m_a^*}
\end{aligned}$$

In every progressive round, except possibly the last, Alice's block size is at least $R_0 2^{-\log_4(1+m_a^*)}$. Thus in all but possibly the last progressive round, Alice always adds bits to her verified transcript at a rate of at least

$$\frac{R_0 2^{-\log_4(1+m_a^*)} - 2F}{R_0 2^{-\log_4(1+m_a^*)}}.$$

Thus, the total number of bits Alices sends in all but the last progressive round is no more than

$$L \cdot \frac{R_0 2^{-\log_4(1+m_a^*)}}{R_0 2^{-\log_4(1+m_a^*)} - 2F}.$$

We will make use of the inequality

$$\frac{1}{1-\delta} \leq 1 + 2\delta \qquad \text{for } 0 < \delta \leq 1/2$$

and let $\delta = 2F/R_0 2^{-\log_4(1+m_a^*)}$. Note that $\delta \leq 1/2$, since Alice's round size is always at least $4F$.

Then we have that the total number of bits sent by Alice in all but the last progressive round is no more than

$$L + \frac{4LF}{R_0 2^{-\log_4(1+m_a^*)}}.$$

Adding in the last progressive round, we get that the total number of bits sent by Alice in progressive rounds is no more than

$$L + \frac{4LF}{R_0 2^{-\log_4(1+m_a^*)}} + R_0 2^{-\log_4(1+m_a^*)}.$$

Putting this together with the number of bits send in non-progressive rounds, we have that the total number of bits send by Alice is no more than

$$\begin{aligned} L + 4R_0\sqrt{m_a^*} + \frac{4LF}{R_0 2^{-\log_4(1+m_a^*)}} + R_0 2^{-\log_4(1+m_a^*)} &\leq L + 5R_0\sqrt{m_a^*} + 4\sqrt{LF}(2^{\log_4(1+m_a^*)}) \\ &\leq L + 10\sqrt{LFm_a^*} + 4\sqrt{LF(1+m_a^*)} \\ &\leq L + 14\sqrt{LF(1+m_a^*)} \end{aligned}$$

□

Lemma 3.22. *Bob terminates in at most* $L + 14\sqrt{LF(1+m_a^*)} + 8\sqrt{LFm_b^*}$ *steps.*

Proof. Since Bob never leaves before Alice, Bob's cost must be at least as much as Alice's. We now compute Bob's additional cost.

At the time of Alice's departure, $r_a = R_0/2^{\lfloor \log_4(1+m_a^*)\rfloor}$. By Lemma 3.13, $r_b \leq 2R_0/2^{\lfloor \log_4(1+m_a^*)\rfloor}$. Let m_b' denote Bob's error count when Alice leaves the algorithm. Then $1 + m_b' \geq 4^{\lfloor \log_4(1+m_a^*)\rfloor - 1}$. Bob's final error count is m_b^*. Thus, Bob's additional cost is at most

$$\begin{aligned}\sum_{i=m_b'}^{m_b^*-1} \frac{R_0}{2^{\lfloor \log_4(1+i) \rfloor}} &\leq 2R_0 \sum_{i=1}^{m_b^*} \frac{1}{2^{\log_4 i}} \\ &= 2R_0 \sum_{i=1}^{m_b^*} \frac{1}{i^2} \\ &\leq 4R_0 \sqrt{m_b^*} \\ &\leq 8\sqrt{LFm_b^*}\end{aligned}$$

Combining this with Alice's cost gives the result. □

Lemma 3.23. *The algorithm ends in at most* $12L$ *time steps.*

Proof. By Lemma 3.22, Bob terminates in at most $L + 14\sqrt{LF(1+m_a^*)} + 8\sqrt{LFm_b^*}$ steps. Moreover, m_a^* and m_b^* are no more than $R_0^2/4F^2 - 1$. Thus, the algorithm terminates in at most the following number of steps.

$$\begin{aligned}L + 14\sqrt{LF(1+m_a^*)} + 8\sqrt{LFm_b^*} &\leq L + 22\sqrt{\frac{LFR_0^2}{4F^2}} \\ &= L + 22\sqrt{\frac{L^2}{4}} \\ &= 12L\,.\end{aligned}$$

□

Lemma 3.24. *If* $T \leq \frac{L}{8F} - 1$ *then both players terminate with the correct output in at most* $L + O(\sqrt{LF(T+1)})$ *steps.*

Proof. Let T_a denote the number of bits flipped by the adversary while Alice is still in the protocol, and T_b the bits flipped after Alice has left. Then $T_a + T_b = T$.

By Lemma 3.20, $m_a^* \leq T_a + \sqrt{T_a}$. By Lemmas 3.3 and 3.18, $m_b^* \leq m_a^* + T_b$. Since $T_a + T_b = T$ it follows that

$$m_a^* \leq T + \sqrt{T} \leq 2T \leq \frac{L}{4F} - 2 < \frac{R_0^2}{4F^2} - 1$$

and similarly

$$m_b^* < \frac{R_0^2}{4F^2} - 1$$

Thus, Alice and Bob will both terminate by outputting the bits of π by Lemma 3.17.

Plugging $m_a^* \leq 2T$ and $m_b^* \leq 3T$ into Lemma 3.22 gives the total number of steps required. □

Lemma 3.25. *With high probability in* L, *there are no hash collisions.*

Proof. By Lemma 3.23, the algorithm ends in at most 12L steps. Also, there are at least $4F = \Theta(\log L)$ steps in a round. Thus, the algorithm has at most $O(L \log L)$ rounds. Each round has one fingerprint. By Theorem 2.1 and the setting of our fingerprint sizes, each fingerprint fails with probability at most $1/L^2$. Thus, a simple union bound gives the result. □

Lemma 3.26. *With high probability in L, any bit flipping of a AMD encoded message is detected.*

Proof. We noted in the previous lemma that the algorithm terminates in $O(L \log L)$ rounds. Each round has two AMD encoded messages. By Theorem 2.2 and the setting of our encoding sizes, each AMD encoding fails with probability at most $1/L^2$. Again, a union bound gives the result. □

4. Unbounded T - Algorithm

Algorithm 1 uses fingerprints of a fixed size, F in order to check its transcripts. Each of these has a $1/L^2$ chance to fail due to a hash collision. Since the algorithm only computes about $O(L/\log L)$ fingerprints, a union bound tells us that with high probability the algorithm succeeds, below its threshold value of T. When T is large, many more checks may need to be made, and eventually there will be a good chance that there is a hash collision. Since the algorithm cannot really recover from a hash collision, we cannot afford this. On the other hand, we cannot simply start out with larger fingerprints, both because this would be too expensive if T turned out to be small, and also because even bigger fingerprints are still of a fixed size and eventually become unreliable. A natural solution is is to allow the fingerprints to grow, adapting the size to the value of T seen so far, and this is indeed what we will do.

4.1. Helper Functions

As in Algorithm 1, we make black-box use of the Naor and Naor hash family, as well as AMD codes to protect information. However, in Iteration j we need the failure probabilities for both these primitives to be $1/(2^j L^2)$. Thus, we want the fingerprint size to grow with j. We will denote the hash function which has a collision probability of at most $1/(2^j L^2)$ by h_j. [4] It is easy to see that $O(j)$ extra bits are required for this, so that the fingerprint size is $O(j + \log L)$.

Algorithm 1 works well when the adversary can only afford to flip a fraction of a bit per block of the algorithm. In this case, it doesn't matter that he can corrupt an entire round of the protocol by flipping a single bit. However, when the adversary has a larger budget, it becomes crucial to force him to pay a larger price to corrupt a round. To this end, we wrap each fingerprint and protocol bit in a linear error-correcting code.

[4] By abuse of notation, we will *not* subscript all the other helper functions with j; it should be clear from context that the version of the function used is the one that operates on strings of the correct size and has the correct failure probability

To be concrete, we will use a repetition code for each protocol bit, and a Reed-Solomon code [22] to provide the already AMD-encoded messages with a degree of error correction. This enables us to encode a message so that it can be recovered even if the adversary corrupts a third of the bits. We will denote the encoding and decoding functions by ecEnc and ecDec respectively. The following theorem, a slight restatement from [22], gives the properties of these functions.

Theorem 4.1. *[22] There is a constant $c > 0$ such that for any message m, $|\operatorname{ecEnc}(m)| \leq c|m|$. Moreover, if m' differs from* $\operatorname{ecEnc}(m)$ *in at most one-third of its bits, then* $\operatorname{ecDec}(m') = m$.

Finally, we observe that the linearity of ecEnc and ecDec ensure that when the error correction is composed with the AMD code, the resulting code has the following properties:

1. If at most a third of the bits of the message are flipped, then the original message can be uniquely reconstructed by rounding to the nearest codeword in the range of ecEnc.
2. Even if an arbitrary set of bits is flipped, the probability of the change not being recognized is at most δ, *i.e.* the same guarantee as the AMD codes.

This is because ecDec is linear, so when noise η is added by the adversary to the codeword x, effectively what happens is the decoding function $\operatorname{ecDec}(x + \eta) = \operatorname{ecDec}(x) + \operatorname{ecDec}(\eta) = m + D(\eta)$, where m is the AMD-encoded message. But now $\operatorname{ecDec}(\eta)$ is an obliviously selected string added to the AMD-encoded codeword.

4.2. Algorithm

Let $N_1 := \lceil 8L/F \rceil$ be the number of rounds in Iteration 1. Let $N_j := 2^{j-1}N_1$ be the number of rounds in Iteration $j > 1$. Let $F_j = 2\beta j + F$ be the size of the fingerprints in Iteration j, where β is the constant from the Naor and Naor hash function. Thus the hash collision probability of a single fingerprint is $2^{-2j}L^{-2}$. Each round of the iteration begins with Alice sending Bob a (1/3)-error-corrected, AMD-encoded synchronization message of length cF_j, followed by simulation of the protocol for F_j channel steps, followed by Bob sending Alice a (1/3)-error-corrected, AMD-encoded fingerprint of length cF_j. Here c is the constant factor blowup we get from the ECC and AMD encodings, but for technical reasons we will further ensure that it is at least 5. Thus, the total round length is $(2c + 1)F_j \geq 11F_j$. We will let α equal $(2c + 1)$.

As in Algorithm 1, Alice will decide whether to update her verified transcript and advance to the next block of π or to rewind to redo the current block, based on whether she receives a fingerprint from Bob that matches the fingerprint of her own transcript. Similarly, Bob will decide whether to join in the simulation of π or to transmit random bits until the end of the round based on receiving or failing to receive Alice's synchronization message at the round's start. Where the round differs from a round in Algorithm 1, is in the actual simulation of π. For the whole iteration, a fixed number of bits of π will be simulated per round.

Algorithm 2: Interactive Communication: Iteration j

```
ALICE'S PROTOCOL

Parameters: $N_j, F_j, \rho_j$;
for $i = 1$ to $N_j$ do
    $\mathcal{F}_a \leftarrow$ ecEnc(amdEnc($|\mathcal{T}_a^*|$));
    Send $\mathcal{F}_a$;
    if $|\mathcal{T}_a^*| < L$ then
        for the next $\lfloor F_j/\rho_j \rfloor$ bits of $\pi$ do
            if sender then
                Send next bit $\rho_j$ times;
                Append to $\mathcal{T}_a$;
            else
                Receive $\rho_j$ bits;
                Append majority bit to $\mathcal{T}_a$;
        end
    else
        Transmit $F_j$ random bits.
    Receive Bob's $cF_j$-bit message, $\mathcal{F}_b'$;
    if IsCodeword($\mathcal{F}_b'$) then
        if $|\mathcal{T}_a^*| \geq L$ then
            Output $\mathcal{T}_a^*[0 : L]$ and
            Terminate;
        $\mathcal{F} \leftarrow$ amdDec($\mathcal{F}_b'$);
        if MatchesFP($\mathcal{F}, \mathcal{T}_a$) then
            // successful round;
            $\mathcal{T}_a^* \leftarrow \mathcal{T}_a$;
    else
        // round failed;
        $\mathcal{T}_a \leftarrow \mathcal{T}_a^*$;
end
```

```
BOB'S PROTOCOL

Parameters: $N_j, F_j, \rho_j$;
for $i = 1$ to $N_j$ do
    if $|\mathcal{T}_b^*| \geq L$ then
        Wait $cF_j$ channel steps;
        Receive $F_j$ bits;
        if fewer than $F_j/3$ alternations in the
        received string then
            Output $\mathcal{T}_b^*[0 : L]$ and
            Terminate;
        else
            $\mathcal{F}_b \leftarrow$ ecEnc(amdEnc(h$_j$($\mathcal{T}_b^*$)));
            Send $\mathcal{F}_b$;
    else
        Receive Alice's $cF_j$-bit message $\mathcal{F}_a'$;
        if IsCodeword(ecDec($\mathcal{F}_a'$)) then
            $\ell \leftarrow$ amdDec(ecDec($\mathcal{F}_a'$));
            if $\ell > |\mathcal{T}_b^*|$ then
                $\mathcal{T}_b^* \leftarrow \mathcal{T}_b$;
            else
                $\mathcal{T}_b \leftarrow \mathcal{T}_b^*$;
            for the next $\lfloor F_j/\rho_j \rfloor$ bits of $\pi$ do
                if sender then
                    Send next bit $\rho_j$ times;
                    Append to $\mathcal{T}_b$;
                else
                    Receive $\rho_j$ bits;
                    Append majority bit to $\mathcal{T}_b$;
            end
            $\mathcal{F}_b \leftarrow$ ecEnc(amdEnc(h$_j$($\mathcal{T}_b$)));
            Send $\mathcal{F}_b$;
        else
            Transmit $(c + 1)F_j$ random bits.
end
```

Algorithm 3: Interactive Communication

```
ALICE'S PROTOCOL

// Iteration 0;
Run Alice's protocol from Alg 1 ;
if not terminated then
    transmit random bits until channel step $12L$;
// End of Iteration 0;
$j \leftarrow 1$;
while still present do
    // Iteration $j$;
    $F_j \leftarrow \beta(j + \log L)$;
    $\rho_j \leftarrow 2^j \lceil \frac{F_j}{F} \rceil \wedge F_j$;
    $N_j \leftarrow 2^{j-1} \lceil 8L/F \rceil$;
    Run Alice's protocol from Algorithm 2, with parameters $N_j, F_j, \rho_j$;
    // End of Iteration $j$;
    $j \leftarrow j + 1$;
end
```

```
BOB'S PROTOCOL

// Iteration 0;
Run Bob's protocol from Alg 1 ;
if not terminated then
    transmit random bits until channel step $12L$;
// End of Iteration 0;
$j \leftarrow 1$;
while still present do
    // Iteration $j$;
    $F_j \leftarrow \beta(j + \log L)$;
    $\rho_j \leftarrow 2^j \lceil \frac{F_j}{F} \rceil \wedge F_j$;
    $N_j \leftarrow 2^{j-1} \lceil 8L/F \rceil$;
    Run Bob's protocol from Algorithm 2, with parameters $N_j, F_j, \rho_j$;
    // End of Iteration $j$;
    $j \leftarrow j + 1$;
end
```

Each bit will be repeated $\rho_j = 2^{j-1}\lceil F_j/F \rceil \wedge F_j$ times. [5] The receiving party will use majority filtering to infer the transmitted bit. Since F_j time steps in the round are allocated to protocol simulation, this allows $\lfloor F_j/\rho_j \rfloor$ bits of π to be simulated.

Notice that the number of rounds doubles from one iteration to the next. Also, the number of repetitions of each simulated bit also roughly doubles between iterations, at least until it hits its cap, which is a constant fraction of the length of the round. This is the so-called doubling trick, (though in our case perhaps it should be quadrupling) which results in the overall cost being dominated by the cost in the last (or second to last) iteration.

5. Unbounded T - Analysis

We now analyze the main algorithm presented in Section 4. As in Section 3, we begin by noting that a hash collision or an AMD code failure will cause the algorithm to fail. Additionally, the algorithm could fail during the padding rounds, if the adversary happens to flip bits in such a way as to cause Alice's random bits to look like silence, resulting in Bob's premature departure.

In Section 5.3 we will show that with high probability each of these events does not occur. Meanwhile, throughout this section we will assume without further mention that we are in the good event where none of the undesirable events occur.

5.1. Alice and Bob are both present

Lemma 5.1. *For every $j \geq 1$, Alice and Bob are always synchronized. That is, they begin the iteration as well as every round therein at the same time.*

Proof. Alice and Bob synchronize themselves after Iteration 0 by both starting Iteration 1 at channel step $12L + 1$. Thereafter, for each $j \geq 1$, they have the same round sizes αF_j and number of rounds N_j in Iteration j, so that they remain synchronized. □

We will call a round *corrupted* if enough bits are flipped in the round that the bits of π being simulated cannot be recovered or verified by Alice. We will call it *uncorrupted* or *progressive* if it is not corrupted in the above sense.

Lemma 5.2. *Each round is either corrupted at a cost of at least $\rho_j/2$ to the adversary or results in $\lfloor F_j/\rho_j \rfloor$ bits of progress in π.*

Proof. Since each simulated protocol bit is sent ρ_j times, with majority filtering at the receiving end, it costs the adversary $\rho_j/2$ to corrupt the repetition-encoded bit. It costs the adversary at least $cF_j/3 \geq \rho_j/2$ to

[5] We remind the reader that $x \wedge y$ denotes the minimum of x and y, while $x \vee y$ denotes their maximum.

corrupt Alice's synchronization message or Bob's fingerprint since these are protected by error-correction. Thus it costs the adversary at least $\rho_j/2$ to corrupt the round. Otherwise, since there are F_j steps allocated to sending protocol bits, and each one is repeated ρ_j times, the protocol is successfully simulated for $\lfloor \frac{F_j}{\rho_j} \rfloor$ bits. □

The following lemma is the equivalent of Lemmas 3.14 to 3.17 for Iteration j. Its proof is nearly identical to the proofs in Section 3.2 (indeed, it is simpler, since Iteration j does not have the synchronization problems faced by Algorithm 1) and we omit it.

Lemma 5.3. *Iteration j has the following properties:*

1. *It is always the case that $\mathcal{T}_a^* \preccurlyeq \pi$, where π is the padded transcript.*
2. *At the beginning and end of each round,*

$$\mathcal{T}_b^* \preccurlyeq \mathcal{T}_a^* = \mathcal{T}_a \preccurlyeq \mathcal{T}_b;$$

where at most one of the inequalities is strict. Moreover, at the end of a channel step in which Bob receives $\mathcal{F}_a$ correctly,

$$\mathcal{T}_b^* = \mathcal{T}_b = \mathcal{T}_a^*.$$

3. *Bob leaves after Alice. When Alice leaves, $|\mathcal{T}_b^*| \geq L$.*
4. *When either party terminates, their output is correct.*

Lemma 5.4. *There are at most $N_j/4$ uncorrupted rounds in Iteration j*

Proof. Since each uncorrupted round results in $\lfloor F_j/\rho_j \rfloor$ bits of progress in π, $\lceil L\rho_j/F_j \rceil$ rounds are sufficient for Alice's transcript length to exceed L. One additional uncorrupted round is sufficient for Bob to catch up to Alice if necessary, using her synchronization message, and for Alice to infer from Bob's fingerprint that Bob's transcript length has exceeded L, resulting in Alice's departure. After that, if a round is uncorrupted, then Bob will perceive silence on the channel, resulting in Bob's departure. Thus $\lceil L\rho_j/F_j \rceil + 2$ uncorrupted rounds are enough for both parties to terminate. Finally note that for all $j \geq 1$,

$$\frac{\rho_j}{F_j} \leq \frac{2^{j-1}}{F} \wedge 1 \leq \frac{2^{j-1}}{F}$$

It follows that (for sufficiently large L) there are at most $2^j L/F = N_j/4$ uncorrupted rounds in Iteration j. □

The following corollary is immediate.

Corollary 5.5. *If j is not the last iteration, then at least $3/4$ of the rounds are corrupted.*

Although the adversary can flip any number of bits in a round, we will only charge him the minimum number of bit-flips required for the outcome we see in the round, *i.e.*, we will charge him 0 for uncorrupted rounds and $\rho_j/2$ for corrupted rounds. Let T_j denote the number of corruptions charged to the adversary in Iteration j. Clearly, for $j > 0$

$$T_j \leq \frac{1}{2} N_j \rho_j \tag{1}$$

Also, we know from Section 2 that if the algorithm does not end in Iteration 0, then $T_0 \geq L/8F$. In this case, we will generously only charge the adversary that amount. In other words, if Iteration 1 is reached, either by both Alice and Bob, or by Bob alone, $T_0 = \lceil L/8F \rceil$.

Lemma 5.6. *If j is not the last iteration then $T_j \geq \frac{3}{8} N_j \rho_j$*

Proof. This follows from Corollary 5.5, since it costs the adversary at least $\rho_j/2$ to corrupt a round. □

Lemma 5.7. *If j is not the last iteration then*

$$3T_{j-1}/2 \leq T_j \leq 64 T_{j-1}$$

Proof. If $j = 1$

$$T_1 \geq \frac{3}{8} N_1 \rho_1 \geq \frac{3L}{F} \geq 24 T_0 > 3T_0$$

and

$$T_1 \leq N_1 \rho_1 / 2 \leq \frac{8L}{F} = 64 T_0 \,.$$

If $j > 1$, then by (1) and Lemma 5.6,

$$\frac{3}{2} \frac{3N_j \rho_j / 8}{N_{j-1} \rho_{j-1}/2} \leq \frac{T_j}{T_{j-1}} \leq \frac{N_j \rho j/2}{3N_{j-1}\rho_{j-1}/8} \leq \; 64$$

since $N_{j-1} = N_j/2$ and $\rho_{j-1} \leq \rho_j \leq 4\rho_{j-1}$. □

Lemma 5.8. *The cost to either player due to uncorrupted rounds in Iteration $j \leq \log F$ is at most*

$$7\alpha \sqrt{L T_{j-1} F}$$

Proof. Each uncorrupted round costs the players αF_j. Since there are at most $N_j/4$ uncorrupted rounds, the resulting cost is no more than $\frac{\alpha}{4} N_j F_j$. Since $j \leq \log F$, $\rho_j = 2^{j-1} \lceil F_j/F \rceil$ and $F_j \leq 2F$. Combining these we have

$$F_j \leq F \sqrt{2^{2-j} \rho_j}$$

so that

$$\begin{aligned}\frac{\alpha}{4}N_jF_j &\le \alpha N_{j-1}F_{j-1}\\ &\le \alpha N_{j-1}F\sqrt{2^{3-j}\rho_{j-1}}\\ &\le \alpha F\sqrt{N_{j-1}2^{3-j}}\sqrt{N_{j-1}\rho_{j-1}}\\ &\le \alpha F\sqrt{2N_1}\sqrt{8T_j/3}\\ &\le \alpha\sqrt{128LT_jF/3}\\ &\le 7\alpha\sqrt{LT_jF}\,. \qquad \square\end{aligned}$$

Lemma 5.9. *If $j > \log F$, the cost to either player due to uncorrupted rounds in Iteration j is at most*

$$3\alpha T_{j-1}$$

Proof. When $j > \log F$, $F_j = \rho_j$ and by Lemma 5.6,

$$\frac{\alpha}{4}N_jF_j = \frac{\alpha}{4}N_j\rho_j \le \alpha N_{j-1}\rho_{j-1} \le \frac{8\alpha}{3}T_{j-1} \le 3\alpha T_{j-1}\,. \qquad \square$$

Lemma 5.10. *The cost to the players from corrupted rounds in Iteration j is at most $4\alpha\sqrt{2LT_jF}$ if $j \le \log F$ and $2\alpha T_j$ otherwise.*

Proof. Suppose there are k corrupted rounds. Then the cost to the players is $k\alpha F_j$, while the adversary's cost is $k\rho_j/2$. If $j \ge \log F + 1$, $F_j = \rho_j$ and we easily see that the players' cost is at most $2\alpha T$. When $j \le \log F$, since $k \le N_j$,

$$\begin{aligned}k\alpha F_j &= \alpha\sqrt{k\rho_jF2^{1-j}}\sqrt{N_jF_j}\\ &\le \alpha\sqrt{T_jF2^{2-j}}\sqrt{2^jN_1F}\\ &\le 2\alpha\sqrt{8LT_jF}\,. \qquad \square\end{aligned}$$

Collecting the various costs and noting that $T_j \le 64T_{j-1}$, we see that for a suitably large constant γ, we have

Lemma 5.11. *The total cost to the players from Iteration j is at most $\gamma\sqrt{LT_{j-1}\log L}$ if $j \le \log F$ and γT_{j-1} otherwise.*

5.2. Bob plays alone

After Alice's verified transcript has length at least L, in each subsequent round, she transmits her synchronization message, and then random bits to indicate her continued presence. Once Alice has left, there is silence on the channel. To corrupt this silence, the adversary must make it look like a corrupted synchronization message followed by random bits. Since a random string of length F_j has, on average, $F_j/2$ alternations of bits, Bob considers the string to represent silence if it has fewer than $F_j/3$ alternations. Thus, to corrupt such a round the adversary must pay at least $F_j/3$.

Alice leaves when she has received word that Bob has a verified transcript of length at least L, and a single extra uncorrupted round thereafter will cause Bob to leave as well. Thus, if iteration j was not Bob's last one, the adversary must have corrupted every round. If $1 \leq k < N_j$ rounds are corrupted, Bob pays at most $(k+1)\alpha F_j \leq 2k\alpha F_j$ and the adversary pays $kF_j/3$. If $k = 0$, we will generously account for the lone uncorrupted round from Iteration j in Iteration $j-1$ by noting that $\alpha(N_{j-1}F_{j-1} + F_j) \leq 2\alpha(N_{j-1}F_{j-1})$ Finally a calculation identical to that in Lemma 5.10 shows that Bob's cost for an iteration j that he played alone is no more than

$$\gamma\sqrt{LT_{j-1}\log L}$$

if $j < \log F$ and

$$\gamma T_{j-1}$$

otherwise.

5.3. Failure Probabilities

In this section we bound the probabilities of the events that cause the algorithm to fail.

Lemma 5.12. *With high probability in L, there is no hash collision during Iteration j.*

Proof. The fingerprint size has been selected large enough that the probability of a hash collision for a single hash is $\frac{1}{2^{2j}L^2}$. Since there are $N_j = 2^{j+2}L/F$ rounds in Iteration j, by a union bound, the probability of a hash collision during the iteration is $O\left(\frac{1}{2^j L \log L}\right)$. □

Lemma 5.13. *With high probability in L, any bit flipping of an AMD encoded message during Iteration j is detected.*

Proof. The size of the AMD encoding has been selected so that the probability of a failure to detect a single instance of tampering is $\frac{1}{2^{2j}L^2}$. Since there are two AMD encodings per round and $2^{j+2}L/F$ rounds, again the probability that such a failure occurs during the iteration is $O\left(\frac{1}{2^j L \log L}\right)$. □

Lemma 5.14. *With high probability in L, Alice leaves before Bob.*

Proof. Bob does not terminate until he thinks Alice has left, and he does not even start checking for whether she seems to have left until after his transcript has length at least L. Since Bob's transcript lags behind that of Alice, this means that by the time Bob is checking for whether Alice has left, Alice either really has left, in which case it is fine for Bob to leave, or she is transmitting i.i.d. random bits in batches of length F_j, between fingerprints. Since the adversary cannot see the bits, any bit flips on his part do not alter the fact that the string received by Bob is a uniformly random bit string of length F_j. Such a string has $F_j/2$ alternations (consecutive bits that differ) in expectation. Bob leaves if he sees fewer than $F_j/3$ alternations. If the string is random, the likelihood of Bob seeing fewer than $F_j/3$ alternations is, by Chernoff's bound, at most $\mathrm{e}^{-F_j/18} \leq \frac{1}{2^{2j}L^2}$ provided $\beta = \frac{F_j}{2j+\log L}$ was chosen suitably large. Since there are at most N_j chances in Iteration j for the adversary to try this attack, a union bound again shows that Bob leaves after Alice, except with probability $O\left(\frac{1}{2^j L \log L}\right)$. □

5.4. Putting everything together

We will now prove our main theorem by putting all these costs together and calculating the total cost to either player and the failure probability of the algorithm. As before, T denotes the number of bits flipped by the adversary.

Theorem 5.15. *The algorithm succeeds with probability at least* $1 - 1/L \log L$. *If it succeeds, then each player's cost is at most*

$$L + O(\sqrt{LT \log L} + T)$$

Proof. First we note that for each $j \geq 0$ (Iteration 0 being Algorithm 1), the probability that Algorithm 3 fails during iteration j is at most $O\left(\frac{1}{2^{2j} L \log L}\right)$. Thus the overall probability that it fails at all is

$$O\left(\sum_{j=0}^{\infty} \frac{1}{2^j L \log L}\right) = O\left(\frac{1}{L \log L}\right)$$

Thus, with high probability the algorithm succeeds.

Let J denote the last iteration in which the player participates. If $J = 0$ then Lemma 3.24 already proves that the players' total cost is at most $L + O(\sqrt{L(T+1)\log L})$. Suppose $J \geq 1$. For each j, let $Cost(j)$ denote the player's cost from Iteration j. We know that

- $Cost(0) = 12L \leq L + \gamma\sqrt{LT_0 \log L}$ where $T_0 = L/(8F)$
- $Cost(j) \leq \gamma\sqrt{LT_{j-1} \log L}$ if $1 \leq j \leq \log F$
- $Cost(j) \leq \gamma T_{j-1}$ if $j > \log F$

When $J \leq \log F$, the player's total cost is

$$\begin{aligned}
\sum_{j=0}^{J} Cost(j) &\leq Cost(0) + \sum_{j=1}^{J} Cost(j) \\
&\leq L + \gamma\sqrt{LT_0 \log L} + \sum_{j=1}^{J} \gamma\sqrt{LT_{j-1}\log L} \\
&\leq L + \gamma\sqrt{L\log L}\left(\sqrt{(2/3)^{J-1}T_{J-1}} + \sum_{j=1}^{J}\sqrt{(2/3)^{J-1-j}T_{J-1}}\right) \\
&\leq L + \gamma\sqrt{LT_{J-1}\log L}\left(\sqrt{(2/3)^{J-1}} + \sum_{j=0}^{J-2}\sqrt{(2/3)^{j}}\right) \\
&\leq L + \frac{\sqrt{3}\gamma}{\sqrt{3}-\sqrt{2}}\sqrt{LT_{J-1}\log L} \\
&= L + \gamma'\sqrt{LT_{J-1}\log L} \\
&\leq L + \gamma'\sqrt{LT\log L}
\end{aligned}$$

On the other hand, $T_{\lfloor \log F \rfloor} = \Theta(N_{\lfloor \log F \rfloor}\rho_{\lfloor \log F \rfloor}) = \Theta(L\log L)$, so that $\sqrt{LT_{\lfloor \log F \rfloor}\log L} = \Theta(T_{\lfloor \log F \rfloor})$ and for $J > \log F$ we have

$$\begin{aligned}
\sum_{j=0}^{J} Cost(j) &\leq Cost(0) + \sum_{j=1}^{\lfloor \log F \rfloor} Cost(j) + \sum_{j=\lfloor \log F \rfloor+1}^{J} Cost(j) \\
&\leq L + \gamma'\sqrt{LT_{\lfloor \log F \rfloor}\log L} + \sum_{j=\lfloor \log F \rfloor+1}^{J} \gamma T_{j-1} \\
&\leq L + \gamma'' T_{\lfloor \log F \rfloor} + \sum_{j=\lfloor \log F \rfloor+1}^{J} \gamma T_{j-1} \\
&\leq L + O(T)
\end{aligned}$$

Thus the players' cost is always $L + O\left(\sqrt{L(T+1)\log L} + T\right)$. □

6. Some Additional Remarks

Need for Private Channels

The following theorem justifies our assumption of private channels.

Theorem 6.1. *Consider any algorithm for interactive communication over a public channel that works with unknown T and always terminates in the noise-free case. Any such algorithm succeeds with probability at most $1/2$.*

Proof. The adversary chooses some protocol π with transcript length L and some separate "corrupted" protocol π_c such that 1) π_C has transcript length L and 2) Bob's individual input for π_c is equivalent to his individual input for π. The goal of the adversary will be to convince Bob that π_c is the protocol, rather than π. Note that we can always choose some appropriate pair π and π_c meeting the above criteria.

Assume that if π_c is the protocol and there is no noise on the channel, then Bob will output π_c with probability at least 1/2; if not, then the theorem is trivially true. Then, the adversary sets π to be the input protocol. Next, the adversary simulates Alice in the case where her input protocol is π_c, and sets the bits received by Bob to be the bits that would be sent by Alice in such a case.

Since the the algorithm eventually terminates, Bob will halt after some finite number of rounds, X. Using the above strategy, Bob will incorrectly output π_c with probability at least 1/2 and the value of T will be no more than X.

Note that in the above, we critically rely on the fact that T is unknown to Bob. □

Communication Rate Comparison.

In Haeupler's algorithm [2], the noise rate ϵ is known in advance and is used to design an algorithm with a communication rate of $1 - O(\sqrt{\epsilon \log\log 1/\epsilon})$. Let L' be the length of π'. Then in his algorithm, $L' = O(L)$, and so the adversary is restricted to flipping $\epsilon L' = O(L)$ bits. Thus, in his model, T and L' are always $O(L)$. In our model, the values of T and L' are not known in advance, and so both T and L' may be asymptotically larger than L.

How do our results compare with [2]? As noted above, a direct comparison is only possible when $T = O(L)$. Restating our algorithm in terms of ϵ, we have the following theorem.

Theorem 6.2. *If the adversary flips $O(L)$ bits and the noise rate is ϵ then our algorithm guarantees a communication rate of* $1 - O\left(\sqrt{\frac{\log L}{L}} + \sqrt{\epsilon \log L}\right)$.

Proof. When $T < L$ we also have $T < \sqrt{L(T+1)\log L}$ and our algorithm guarantees that for some $\gamma > 0$,

$$L' = L + \gamma\sqrt{L(T+1)\log L}$$

Let $\epsilon = T/L'$ and $R = L/L'$ be the effective noise and communication rates respectively. Then,

$$\begin{aligned}
R = \frac{L}{L'} &= 1 - \frac{L' - L}{L'} \\
&\geq 1 - \frac{\gamma\sqrt{L(T+1)\log L}}{L'} \\
&\geq 1 - \gamma\frac{\sqrt{L\log L} + \sqrt{LT\log L}}{L'} \\
&\geq 1 - \gamma\left(\frac{\sqrt{R\log L}}{\sqrt{L'}} + \sqrt{R\epsilon\log L}\right) \\
&\geq 1 - \gamma\sqrt{\log L}\left(\frac{1}{\sqrt{L}} + \sqrt{\epsilon}\right),
\end{aligned}$$

where the last line follows because $1/\sqrt{L'} \leq 1/\sqrt{L}$ and $R \leq 1$. □

We note that the additive term $\sqrt{\frac{\log L}{L}}$ arises from the fact that because we do not know the error rate ahead of time, we cannot get a communication rate of 1 even when the effective error rate turns out to be zero.

A Note on Fingerprint Size.

A natural question is whether more powerful probabilistic techniques than union bound could enable us to use smaller fingerprints as done in [2]. The variability of block sizes poses a challenge to this approach since Alice and Bob must either agree on the current block size, or be able to recover from a disagreement by having Bob stay in the listening loop so he can receive Alice's message. If their transcripts diverge by more than a constant number of blocks, it may be difficult to make such a recovery, and therefore it seems challenging to modify our algorithm to use smaller fingerprints. However, it is a direction for further investigation.

A Lower Bound

In this section, we prove a lower bound that demonstrates the near optimality of our upper bound by assuming the following conjecture by Haeupler holds [2]. We now restate Haeupler's conjecture.

Conjecture 1. *(Haeupler [2], 2014) The maximal rate achievable by an interactive coding scheme for any binary error channel with random or oblivious errors is $1 - \Theta(\sqrt{\epsilon})$ for a noise rate $\epsilon \to 0$. This also holds for for fully adversarial binary error channels if the adversary is computationally bounded or if parties have access to shared randomness that is unknown to the channel.*

For the remainder of this section, we ***assume that Haeupler's conjecture holds*** for any algorithm that succeed with high probability in L with an expected cost of at most L' under adversarial noise. For ease of exposition, we omit such statements in all of our claims below. By *robust* interactive communication, we mean interactive communication tolerates T errors.

We begin by showing the near optimality with respect to the communication rate achieved:

Theorem 6.3. *Any algorithm for robust interactive communication must have $L' = L + \Omega\left(T + \sqrt{LT}\right)$ for some $T \geq 1$.*

Proof. Let $T \geq 1$ be any value such that $T/L' = o(1)$. Then, Haeupler's Conjecture applies and the expected total number of bits sent is $L' \geq L/(1-d\sqrt{\epsilon})$ for some constant $d > 0$. Noting that $1/(1-d\sqrt{\epsilon}) \geq 1+d\sqrt{\epsilon}$ by the well-known sum of a geometric series, this implies that $L' \geq L/(1-d\sqrt{\epsilon}) \geq (1+d\sqrt{\epsilon})L = (1+d\sqrt{T/L'})L$ since $\epsilon = T/L'$.

This implies that $L/L' \leq 1/(1 + d\sqrt{T/L'})$. Now observe that $1/(1 + x) = 1/(1 - (-x)) \leq 1 - x + x^2$ for $|x| < 1$, again by the sum of a geometric series. Plugging in $d\sqrt{T/L'}$ for x, we have $1/(1 + d\sqrt{T/L'}) \leq$

$1-d\sqrt{T/L'}+d^2(T/L')$. Therefore, $L/L' \leq 1-d\sqrt{T/L'}+d^2(T/L') = 1-d\sqrt{T/L'}(1-d\sqrt{T/L'}) \leq 1-d'\sqrt{T/L'}$ for some $d' > 0$ depending only on d.

We then derive: $L \leq L'(1-d'\sqrt{T/L'}) = L' - d'\sqrt{L'T}$. It follows that $L' \geq L + d'\sqrt{L'T} = L + \Omega(\sqrt{LT})$ since $L' \geq L$.

Finally, we show that $\sqrt{LT} = \Theta(T + \sqrt{LT})$. Assume that given any algorithm A for interactive computation, we create a new algorithm A' that has expected value of $L' = O(L)$. To do this, A' checks based on ϵ and L whether or not Haeupler's algorithm [2] will send fewer bits in expectation than A. If so it runs Haeupler's algorithm. Note that the expected number of bits sent by A' is no more than the expected number of bits sent by A.

Note that $T = \epsilon L'$ and for algorithm A', the expected value of $L' = O(L)$. This implies that implies that $T = \epsilon O(L)$ or $T = O(L)$. Since $T < L$, it holds that $\sqrt{LT} = \Theta(T + \sqrt{LT})$ which completes the proof. □

7. Conclusion

We have described the first algorithm for interactive communication that tolerates an unknown but finite amount of noise. Against an adversary that flips T bits, our algorithm sends $L + O\left(\sqrt{L(T+1)\log L} + T\right)$ bits in expectation where L is the transcript length of the computation. We prove this is optimal up to logarithmic factors, assuming a conjectured lower bound by Haeupler. Our algorithm critically relies on the assumption of a private channel, an assumption that we show is necessary in order to tolerate an unknown noise rate.

Several open problems remain including the following. First, can we adapt our results to interactive communication that involves more than two parties? Second, can we more efficiently handle an unknown amount of stochastic noise? Finally, for any algorithm, what are the optimal tradeoffs between the overhead incurred when $T = 0$ and the overhead incurred for $T > 0$?

Acknowledgments

We are grateful to Nico Döttling, Bernhard Haeupler, Mahdi Zamani, and the anonymous reviewers for their useful discussions and comments.

References

[1] V. Dani, T. Hayes, M. Mohavedi, J. Saia, M. Young, Interactive Communication with Unknown Noise Rate, in: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP), 2015, pp. 575–587.

[2] B. Haeupler, Interactive channel capacity revisited, in: Foundations of Computer Science (FOCS), IEEE, 2014, pp. 226–235.

[3] L. Schulman, Communication on Noisy Channels: A Coding Theorem for Computation, in: Foundations of Computer Science (FOCS), 1992, pp. 724–733.

[4] Z. Brakerski, M. Naor, Fast Algorithms for Interactive Coding, in: Symposium on Discrete Algorithms (SODA), 2013, pp. 443–456.

[5] Z. Brakerski, Y. T. Kalai, Efficient Interactive Coding against Adversarial Noise, in: Foundations of Computer Science (FOCS), 2012, pp. 160–166.

[6] M. Braverman, A. Rao, Towards Coding for Maximum Errors in Interactive Communication, in: Symposium on Theory of Computing (STOC), 2011, pp. 159–166.

[7] M. Braverman, Towards Deterministic Tree Code Constructions, in: Innovations in Theoretical Computer Science Conference (ITCS), 2012, pp. 161–167.

[8] R. Gelles, A. Moitra, A. Sahai, Efficient and Explicit Coding for Interactive Communication, in: Foundations of Computer Science (FOCS), 2011, pp. 768–777.

[9] M. Ghaffari, B. Haeupler, M. Sudan, Optimal Error Rates for Interactive Coding I: Adaptivity and Other Settings, in: Symposium on Theory of Computing (STOC), 2014, pp. 794–803.

[10] M. Ghaffari, B. Haeupler, Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding, available at: `http://arxiv.org/abs/1312.1763` (2013).

[11] C. E. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal 27 (3) (1948) 379–423.

[12] L. J. Schulman, Deterministic Coding for Interactive Communication, in: Symposium on Theory of Computing (STOC), 1993, pp. 747–756.

[13] M. Braverman, Coding for Interactive Computation: Progress and Challenges, in: Communication, Control, and Computing (Allerton), 2012, pp. 1914–1921.

[14] M. Peczarski, An Improvement of the Tree Code Construction, Information Processing Letters 99 (3) (2006) 92–95.

[15] C. Moore, L. J. Schulman, Tree Codes and a Conjecture on Exponential Sums, in: Innovations in Theoretical Computer Science (ITCS), 2014, pp. 145–154.

[16] R. Ostrovsky, Y. Rabani, L. J. Schulman, Error-Correcting Codes for Automatic Control, IEEE Transactions on Information Theory 55 (7) (2009) 2931–2941.

[17] M. Franklin, R. Gelles, R. Ostrovsky, L. Schulman, Optimal Coding for Streaming Authentication and Interactive Communication, IEEE Transactions on Information Theory 61 (1) (2015) 133–145.

[18] M. Braverman, K. Efremenko, List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise, in: Foundations of Computer Science (FOCS), 2014, pp. 236–245.

[19] O. Feinerman, B. Haeupler, A. Korman, Breathe before speaking: efficient information dissemination despite noisy, limited and anonymous communication, in: Principles of Distributed Computing (PODC), ACM, 2014, pp. 114–123.

[20] J. Naor, M. Naor, Small-bias probability spaces: Efficient constructions and applications, SIAM Journal on Computing (SICOMP) 22 (4) (1993) 838–856.

[21] R. Cramer, Y. Dodis, S. Fehr, C. Padró, D. Wichs, Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors, in: Advances in Cryptology–EUROCRYPT 2008, Springer, 2008, pp. 471–488.

[22] I. S. Reed, G. Solomon, Polynomial codes over certain finite fields, Journal of the Society for Industrial and Applied Mathematics 8 (2) (1960) 300–304. arXiv:http://dx.doi.org/10.1137/0108018, doi:10.1137/0108018.
URL http://dx.doi.org/10.1137/0108018