Secure one-way interactive communication

Abhinav Aggarwal
abhiag@cs.unm.edu
Varsha Dani
varsha@cs.unm.edu
Thomas Hayes
hayes@cs.unm.edu
Jared Saia
saia@cs.unm.edu

Abstract

Alice and Bob are connected via a two-way binary channel. This paper describes an algorithm to enable Alice to send a message to Bob when 1) an oblivious adversary flips an unknown number of bits, $T$, on the channel; and 2) the message length $L$, and a desired error probability, $\epsilon$ are public knowledge.

With probability at least $1 - \epsilon$, our algorithm ensures that Bob receives the correct message, and that Alice and Bob terminate after sending a total of $L + O\left( T + \min\left( T + 1, \frac{T}{\log T} \right) \log \left( \frac{L}{\epsilon} \right) \right)$ bits.

When $\epsilon = \Omega\left( \frac{1}{\text{poly}(L)} \right)$ and $T$ is large, the number of bits sent is $L + O(T)$, which is asymptotically optimal, assuming a conjecture from [14].

1 Introduction

What if we want to tune the cost of sending a message over a noisy two-way channel to depend on the amount of noise, even when the noise rate is unknown? In particular, consider the case where Alice wants to send a message of length $L$ to Bob, over a two-way channel, even in the presence of an oblivious adversary that can flip an unknown number of bits, $T$, on the channel. We assume that $L$, along with a desired error probability $\epsilon \in (0, 1)$ are public knowledge. We assume that the oblivious adversary knows our algorithm, but does not know the private random bits of Alice and Bob, or the bits sent on the channel, except when these bits do not depend on the private random bits of the algorithm. Our main result is summarized in the following theorem.

**Theorem 1.1.** Our algorithm tolerates an unknown number of adversarial errors, $T$, and for a given $\epsilon \in (0, 1)$, succeeds with probability at least $1 - \epsilon$, and sends $L + O\left( T + \min\left( T + 1, \frac{T}{\log T} \right) \log \left( \frac{L}{\epsilon} \right) \right)$ bits.

For the case where $\epsilon = \Omega\left( \frac{1}{\text{poly}(L)} \right)$ and $T > L$, the algorithm sends $L + O(T)$ bits. This is asymptotically optimal for any algorithm dealing with the case, assuming a conjecture from [14].

1.1 Related Work

**Interactive Communication** The problem of interactive communication asks how two parties can run a protocol $\pi$ over a noisy channel. This problem was first posed by Schulman [25, 24], who describes 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 [3] for an excellent survey).

Schulman’s scheme tolerates an adversarial noise rate of $1/240$, even if the adversary is not oblivious. 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 [4, 22, 18]. There

1
were also efforts to create alternative codes \cite{11, 20}. Recently, elegant computationally-efficient schemes that tolerate a constant adversarial noise rate have been demonstrated \cite{1, 12}. Additionally, a large number of results have improved the tolerable adversarial noise rate \cite{2, 6, 13, 11, 5}, as well as tuning the communication costs to a known, but not necessarily constant, adversarial noise rate \cite{14}.

Our paper builds on a result on interactive communication by Dani et al \cite{8}, which in contrast to previous work, assumes an oblivious adversary, but tolerates an unknown number of bit flips by such an adversary. Their algorithm sends $L + O\left(\sqrt{L(T + 1)\log L + T}\right)$ bits in expectation. They show that the adversary must be oblivious in order to tolerate unknown $T$.

We note that the problem in this paper can be viewed as a special case of interactive communication in the model from \cite{8}, where the protocol $\pi$ just sends $L$ bits from Alice to Bob. However, since the problem in this paper is simpler, we can achieve better communication costs. In particular, when $\epsilon = \Omega\left(\frac{1}{\text{poly}(L, T)}\right)$, our algorithm requires $L + O(\min((T + 1)\log L, L) + T)$ bits. This is never worse than \cite{8}, and can be significantly better. For example, when $T = O(1)$, our cost is $L + O(\log L)$ versus $L + O(\sqrt{L\log L})$ from \cite{8}. In general if $T = o\left(L / \log L\right)$ our cost is asymptotically better than \cite{8}.

\textbf{Rateless Codes} Rateless error correcting codes enable generation of potentially an infinite number of encoding symbols from a given set of source symbols with the property that given any subset of a sufficient number of encoded symbols (typically of the order of the number of source symbols), the original source symbols can ideally be recovered. Two examples of such codes are the Fountain codes \cite{17} and the LT codes \cite{21, 16, 15}. Both of these belong to the class of erasure codes, enabling the recovery of the original message from only a subset of the encoded symbols.

Erasure codes do not usually employ any feedback mechanisms other than for stopping transmission \cite{21, 16} and for error detection \cite{15} at the receiver. Additionally, the feedback channel is typically assumed to be noise free. We differ from this model in that we allow the adversary to flip bits on the feedback channel. Additionally, our model of channel noise in both directions is more challenging than is typically assumed for rateless codes. In particular, adversarial bit flips are more difficult to tolerate than erasures.

\subsection*{1.2 Formal Model}

Our algorithm takes as input the message $M$, which is a sequence of $L$ bits to be sent from Alice to Bob, and an error tolerance $\epsilon > 0$. We assume that both Alice and Bob know $L$ as well as $\epsilon$.

\textbf{Channel steps} We assume that the communication over the channel is synchronous and individual computation is instantaneous. A \textit{channel step} is defined as the amount of time that it takes to send one bit over the channel.

\textbf{Silence on the channel} Similar to \cite{8}, 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 the subsequent steps of the sequence remains the same, unless the adversary pays for one bit flip each time it wants to change the value of the bit received in any contiguous sequence of silent steps.

\subsection*{1.3 Paper organization}

This paper is organized as follows. Section 2 presents our main algorithm. Section 3 presents the analysis for failure probability, correctness, termination and number of bits sent by the algorithm. Finally, Section 4 concludes the paper by stating the main result and some open problems.
2 Algorithm

Our algorithm makes critical use of Reed-Solomon codes from [23]. Alice begins by encoding her message using a polynomial of degree \( d = \lceil L / \log q \rceil - 1 \) over \( GF(q) \), where \( q = 2^{\lceil \log L \rceil} \). She sends the values of this polynomial computed at certain elements of the field as message symbols to Bob. Upon receiving an appropriate number of these points, Bob computes the polynomial using the Berlekamp-Welch algorithm [26] and sends a fingerprint of his guess to Alice. Upon hearing this fingerprint, if Alice finds no errors, she echoes the fingerprint back to Bob, upon receiving a correct copy of which, Bob terminates the algorithm. Unless the adversary corrupts many bits, Alice terminates soon after.

However, in the case where Alice does not receive a correct fingerprint of the polynomial from Bob, she sends two more evaluations of the polynomial to Bob. Bob keeps receiving extra evaluations and recomputing the polynomial until he receives the correct fingerprint echo from Alice.

2.1 Notation

Some helper functions and notation used in our algorithm are described in this section. We denote by \( s \in u.a.r. \) \( S \) the fact that \( s \) is sampled uniformly at random from the set \( S \).

Fingerprinting For fingerprinting, we use a well known theorem by Naor and Naor [19], slightly reworded as follows:

**Theorem 2.1.** [19] Fix integer \( \ell > 0 \) and real \( p \in (0,1) \). Then there exist constants \( C_s, C_h > 0 \) and algorithm \( h \) such that the following hold for a given string \( s \in u.a.r. \{0,1\}^{C_s \log(1/p)} \).

1. For a string \( m \) of length at most \( \ell \), we have \( h(s,m,p,\ell) = (s,f) \), where \( f \) is a string of length \( C_h \log(1/p) \).

2. For any bit strings \( m \) and \( m' \) of length at most \( \ell \), if \( m = m' \), then \( h(s,m,p,\ell) = h(s,m',p,\ell) \), else \( \Pr\{h(s,m,p,\ell) = h(s,m',p,\ell)\} \leq p \).

We refer to \( h(s,m,p,\ell) \) as the fingerprint of the message \( m \).

GetPolynomial Let \( \mathcal{M} \) be a multiset of tuples of the form \( (x,y) \in GF(q) \times GF(q) \). For each \( x \in GF(q) \), we define \( \text{maj}(\mathcal{M})(x) \) to be the tuple \((x,z)\) that has the highest number of occurrences in \( \mathcal{M} \), breaking ties arbitrarily. We define \( \text{maj}(\mathcal{M}) = \bigcup_{x \in GF(q)} \{(x,\text{maj}(\mathcal{M})(x))\} \). Given the set \( S = \text{maj}(\mathcal{M}) \), we define the function \( \text{GetPolynomial}(S,d,q) \) which returns the degree-\( d \) polynomial over \( GF(q) \) that is supported by the largest number of points in \( S \), breaking ties arbitrarily.

The following theorem from [23] [26] provides conditions under which \( \text{GetPolynomial}(S,d,q) \) reconstructs the required polynomial.

**Theorem 2.2.** [23] [26] Let \( P \) be a polynomial of degree \( d \) over some field \( \mathbb{F} \), and \( S \subset \mathbb{F} \times \mathbb{F} \). Let \( g \) be the number of elements \((x,y) \in S\) such that \( y = P(x) \), and let \( b = |S| - g \). Then, if \( g > b + d \), we have \( \text{GetPolynomial}(S,d,q) = P \).

Algebraic Manipulation Detection Codes Our algorithm also makes use of Algebraic Manipulation Detection (AMD) codes from [7]. For a given \( \delta > 0 \), called the strength of AMD encoding, these codes provide three functions: \( \text{amdEnc}, \text{amdDec} \) and \( \text{IsCodeword} \). The function \( \text{amdEnc}(m,\delta) \) creates an AMD encoding of a message \( m \). The function \( \text{IsCodeword}(m,\delta) \) takes a message \( m \) and returns true if and only if there exists some message \( m' \) such that \( \text{amdEnc}(m',\delta) = m \). The function \( \text{amdDec}(m,\delta) \) takes a message \( m \) such that \( \text{IsCodeword}(m,\delta) \) and returns a message \( m' \) such that \( \text{amdEnc}(m',\delta) = m \). These functions enable detection of bit corruption in an encoded message with high probability. The following (slightly reworded) theorem from [7] helps establish this:
Theorem 2.3. [7] For any \( \delta > 0 \), there exist functions \( \text{amdEnc} \), \( \text{amdDec} \) and \( \text{IsCodeword} \), such that for any bit string \( m \) of length \( x \):

1. \( \text{amdEnc}(m, \delta) \) is a string of length \( x + C_a \log(1/\delta) \), for some constant \( C_a > 0 \)

2. \( \text{IsCodeword}(\text{amdEnc}(m, \delta), \delta) \) and \( \text{amdDec}(\text{amdEnc}(m, \delta), \delta) = m \)

3. For any bit string \( s \neq 0 \) of length \( x \), we have \( \Pr(\text{IsCodeword}(\text{amdEnc}(m, \delta) \oplus s, \delta)) \leq \delta \)

With the use of Naor-Naor hash functions along with AMD codes, we are able to provide the required security for messages with Alice and Bob. Assume that the Bob generates the fingerprint with the use of Naor-Naor hash functions along with AMD codes, we are able to provide the required security for messages with Alice and Bob. Assume that the Bob generates the fingerprint along with AMD codes, we are able to provide the required security for messages with Alice and Bob. Assume that the Bob generates the fingerprint, is converted to \( (s \oplus t_1, f \oplus t_2) \) for some strings \( t_1, t_2 \) of appropriate lengths. Upon receiving this, Alice compares it against the fingerprint of her message for appropriately chosen \( p \). Then, we require that there exist a \( \delta \geq 0 \) such that for any choice of \( t_1, t_2 \),

\[
\Pr(h(s \oplus t_1, m', p, |m'|) = (s \oplus t_1, f \oplus t_2)) \leq \delta
\]

for any string \( m' \neq m \). Theorem 2.3 provides us with this guarantee directly.

Error-correction Codes These codes enable 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 \( \text{ecEnc} \) and \( \text{ecDec} \), respectively. The following theorem, a slight restatement from [23], gives the properties of these functions.

Theorem 2.4. [23] There is a constant \( C_e > 0 \) such that for any message \( m \), we have \( |\text{ecEnc}(m)| \leq C_e|m| \). Moreover, if \( m' \) differs from \( \text{ecEnc}(m) \) in at most one-third of its bits, then \( \text{ecDec}(m') = m \).

Finally, we observe that the linearity of \( \text{ecEnc} \) and \( \text{ecDec} \) 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 \( \text{ecEnc} \).

2. Even if an arbitrary set of bits is flipped, the probability of the change not being recognized is at most \( \delta \), i.e. the same guarantee as the AMD codes.

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

Silence In our algorithm, silence on the channel has a very specific meaning. We define the function \( \text{IsSilence}(s) \) to return true iff the string \( s \) has fewer than \(|s|/3 \) bit alternations.

Other notation We use \( 0_b \) to denote the \( b \)-bit string of all zeros, \( \odot \) for string concatenation and \( \text{Listen}(b) \) to denote the function that returns the bits on the channel over the next \( b \) time steps. For the sake of convenience, we will use \( \log x \) to mean \( \lceil \log_2 x \rceil \), unless specified otherwise. Let \( C = \max\{19, C_h + C_a + C_e C_s \} \).

2.2 Algorithm overview

We now present our main algorithm: Algorithm 1 is what Alice follows and Algorithm 2 is what Bob follows. Both algorithms share the knowledge of message length \( L \) and the error tolerance \( \epsilon \).
Algorithm 1 Alice’s algorithm

1: procedure ALICE($M, \epsilon$) \quad \triangleright M is a message of length $L$
2: \hfill $q \leftarrow 2^{\lceil \log L \rceil}$ \quad \triangleright Field size
3: \hfill $d \leftarrow \lceil L / \log q \rceil - 1$ \quad \triangleright Degree of polynomial
4: $P_a \leftarrow$ degree-$d$ polynomial encoding of $M$ over $GF(q)$
5: Send \{$P(0), P(1), \ldots, P(d+1)$\}
6: for $j = 1$ to $\infty$ do \quad \triangleright Rounds for the algorithm
7: \hfill $\delta_j \leftarrow (1/2)^{\lceil j/d \rceil} \epsilon / 6d$
8: \hfill $b_j \leftarrow C \log (L/\delta_j)$ \quad \triangleright Message size in this round
9: \hfill $f \leftarrow$ ecDec (Listen ($b_j$)) \quad \triangleright Fingerprint from Bob
10: if IsCodeword ($f, \delta_j$) then
11: \hfill $(s, f_1) \leftarrow$ amdDec ($f, \delta_j$)
12: \hfill if ($s, f_1 = h(s, P_a, \delta_j, L)$ then
13: \hfill Send ecEnc ($f$) \quad \triangleright Echo the fingerprint
14: \hfill Send 0$_b$, if the fingerprint was not echoed.
15: \hfill $f_2 \leftarrow$ Listen ($b_j$)
16: \hfill if IsSilence ($f_2$) then
17: \hfill Terminate \quad \triangleright Bob has likely left
18: \hfill else
19: \hfill $M_a \leftarrow$ polynomial evaluation tuples of $P_a$ at next two points of the field (cyclically)
20: \hfill Send ecEnc (amdEnc ($M_a, \delta_j$))

Algorithm 2 Bob’s algorithm

1: procedure BOB($L, \epsilon$)
2: \hfill $q \leftarrow 2^{\lceil \log L \rceil}$ \quad Field size
3: \hfill $d \leftarrow \lceil L / \log q \rceil - 1$ \quad Degree of polynomial
4: \hfill $B \leftarrow \emptyset$ \quad $B \in GF(q) \times GF(q)$
5: Listen to first $d + 1$ evaluations from Alice
6: Add the corresponding polynomial evaluation tuples to $B$
7: for $j = 1$ to $\infty$ do \quad Message size in this round
8: \hfill $\delta_j \leftarrow (1/2)^{\lceil j/d \rceil} \epsilon / 6d$
9: \hfill $b_j \leftarrow C \log (L/\delta_j)$
10: \hfill $P_b \leftarrow$ GetPolynomial (maj($B$), $d, q$)
11: \hfill Sample a string $s \in $ u.a.r. \{$0, 1$\}$C, b_j / C$
12: \hfill $f_b \leftarrow$ amdEnc ($h(s, P_b, \delta_j, L), \delta_j$)
13: \hfill Send ecEnc ($f_b$) \quad \triangleright Send Alice the fingerprint of the polynomial
14: \hfill $f'_b = $ ecDec (Listen ($b_j$)) \quad \triangleright Listen to Alice’s echo
15: \hfill if $f'_b = f_b$ then
16: \hfill Terminate
17: \hfill else
18: \hfill Send a string $f'_2 \in $ u.a.r. \{$0, 1$\}$b_j$
19: \hfill Receive polynomial evaluation tuples for the next two field elements and add to $B$
3 Analysis

We now prove that our algorithm is correct with probability at least $1 - \epsilon$, and compute the number of bits sent. Before proceeding to the proof, we define three bad events:

1. **Unintentional Silence.** When Bob executes step $18$ of his algorithm, the string received by Alice is interpreted as silence.

2. **Fingerprint Error.** Fingerprint hash collision as per Theorem 2.1.

3. **AMD Error.** The adversary corrupts an AMD encoded message into an encoding of a different message.

**Rounds** For both Alice and Bob, we define a **round** as one iteration of the for loop in our algorithm. We refer to the part of the algorithm before the for loop begins as **round** 0. The AMD encoding strength $\delta$ is equal to $\epsilon/6d$ initially and decreases by a factor of 2 every $d$ rounds. This way, the number of bits added to the messages increases linearly every $d$ rounds, which enhances security against corruption.

3.1 Correctness and Termination

We now prove that with probability at least $1 - \epsilon$, Bob terminates the algorithm with the correct guess of Alice’s message.

3.1.1 Unintentional Silence

The following lemmas show that Alice terminates before Bob with probability at most $\epsilon/3$.

**Lemma 3.1.** For $b \geq 71$, the probability that a $b$-bit string sampled uniformly at random from $\{0, 1\}^b$ has fewer than $b/3$ bit alternations is at most $e^{-b/19}$.

**Proof.** Let $s$ be a string sampled uniformly at random from $\{0, 1\}^b$, where $b \geq 71$. Denote by $s[i]$ the $i^{th}$ bit of $s$. Let $X_i$ be the indicator random variable for the event that $s[i] \neq s[i+1]$, for $1 \leq i < b$. Note that all $X_i$’s are mutually independent. Let $X$ be the number of bit alternations in $s$. Clearly, $X = \sum_{i=1}^{b-1} X_i$, which gives $E(X) = \sum_{i=1}^{b-1} E(X_i)$, using the linearity of expectation. Since $E(X_i) = 1/2$ for all $1 \leq i < b$, we get $E(X) = (b-1)/2$. Using the multiplicative version of Chernoff bounds \cite{prob}, for $0 \leq t \leq \sqrt{b-1}$,

$$
\Pr \left\{ X < \frac{b-1}{2} - \frac{t\sqrt{b-1}}{2} \right\} \leq e^{-t^2/2}.
$$

To obtain $\Pr\{ X < b/3 \}$, set $t = \frac{b-3}{3\sqrt{b-1}}$ to get,

$$
\Pr\{ X < b/3 \} \leq e^{-\frac{(b-3)^2}{18(b-1)}} \leq e^{-b/19} \quad \text{for } b \geq 71.
$$

**Lemma 3.2.** Alice terminates the algorithm before Bob with probability at most $\epsilon/3$.

**Proof.** Let $\xi$ be the event that Alice terminates before Bob. This happens when the string sent by Bob in step 18 after possible adversarial corruptions is interpreted as silence by Alice. Let $\xi_j$ be the event that Alice terminates before Bob in round $j$ of the algorithm. Then, using a union bound over the rounds, the fact that $C \geq 19$ and Lemma 5.1 we get

$$
\Pr(\xi) \leq \sum_{j \geq 1} \Pr(\xi_j) \leq \sum_{j \geq 1} e^{-b_j/19} \leq \sum_{j \geq 1} 2^{-b_j/19} = \sum_{j \geq 1} 2^{-C \log(L/\delta_j)/19} \leq \sum_{j \geq 1} 2^{-\log(L/\delta_j)} = \sum_{j \geq 1} \log(\delta_j/L) \leq \frac{\epsilon}{6Ld} \sum_{j \geq 0} \left( \frac{1}{2} \right)^{\frac{j}{d}} \frac{1}{2} \leq \frac{\epsilon}{3L} \leq \frac{\epsilon}{3}.
$$

6
Note that Lemma 3.1 is applicable here because for each \( j \geq 1 \), we have \( b_j \geq 71 \). To see this, use the fact that \( d \leq 2L/\log L \) and \( \epsilon < 1 \) to obtain the condition \( L^2 \geq 2^{71/C}/12 \), which is always true because \( L^2 > 4 > 2^{71/C}/12 \).

### 3.1.2 Fingerprint Failure

The following lemma proves that the fingerprint error happens with probability at most \( \epsilon/3 \), ensuring the correctness of the algorithm.

**Lemma 3.3.** Upon termination, Bob does not have the correct guess of Alice’s message with probability at most \( \epsilon/3 \).

**Proof.** Let \( \xi \) be the event that Bob does not have the correct guess of Alice’s message upon termination. Note that in round \( j \), from Theorem 2.1, the fingerprints fail with probability at most \( \delta_j \). Using a union bound over these rounds, we get

\[
\Pr\{\xi\} \leq \sum_{j \geq 1} \delta_j = \sum_{j \geq 1} \frac{\epsilon}{6d} \left(\frac{1}{2}\right)^{\lfloor j/d \rfloor} \leq \frac{\epsilon}{6} \sum_{j \geq 0} \left(\frac{1}{2}\right)^j = \frac{\epsilon}{3}.
\]

### 3.1.3 AMD Failure

**Lemma 3.4.** The probability of AMD failure is at most \( \epsilon/3 \).

**Proof.** Note that in round \( j \), from Theorem 2.3, AMD failure occurs with probability at most \( \delta_j \). Hence, using a union bound over the rounds, the AMD failure occurs with probability

\[
\sum_{j \geq 1} \delta_j = \sum_{j \geq 1} \frac{\epsilon}{6d} \left(\frac{1}{2}\right)^{\lfloor j/d \rfloor} \leq \frac{\epsilon}{6} \sum_{j \geq 0} \left(\frac{1}{2}\right)^j = \frac{\epsilon}{3}.
\]

### 3.2 Probability of Failure

**Lemma 3.5.** Our algorithm succeeds with probability at least \( 1 - \epsilon \).

**Proof.** Lemmas 3.2, 3.3 and 3.4 ensure that the three bad events, as defined previously, each happen with probability at most \( \epsilon/3 \). Hence, using a union bound over the occurrence of these three events, the total probability of failure of the algorithm is at most \( \epsilon \). If the three bad events do not occur, then Alice will continue to send evaluations of the polynomial until Bob has the correct message. Since \( T \) is finite, Bob will eventually have the correct message and terminate.

### 3.3 Cost to the algorithm

Recall that Alice and Bob compute their polynomials \( P_a \) and \( P_b \), respectively, over \( GF(q) \). We refer to every \((x, y) \in GF(q) \times GF(q)\) that Bob stores after receiving the evaluation \( y \), that has potentially been tampered with, of the polynomial \( P_a \) at \( x \) from Alice as a *polynomial evaluation tuple*. We call a polynomial evaluation tuple \((x, y)\) in Bob’s set \( B \) *good* if \( P_a(x) = y \) and *bad* otherwise.

We begin by stating two important lemmas that relate the number of bits flipped by the adversary to make \( m \) polynomial evaluation tuples bad to the number of bits required to send them.

**Lemma 3.6.** Let \( f(m) \) be the number of bits flipped by the adversary to make \( m \) polynomial evaluation tuples bad. Then,

\[
f(m) \geq \begin{cases} m & \text{if } m \leq d + 1 \\ (d + 1) + \frac{\epsilon}{6} \left( (m - d - 1) \log(6Ld/\epsilon) + \frac{(m-d-3)^2}{4d} \right) & \text{otherwise} \end{cases}
\]
Proof. Let \( m = m_1 + m_2 \), where \( m_1 \leq d + 1 \) is the number of polynomial evaluation tuples that were not encoded and \( m_2 \) is the number of AMD and error-encoded polynomial evaluation tuples. Clearly, \( f(m_1) = m_1 \). Each of the remaining \( m_2 \) polynomial evaluation tuples are sent in pairs, one pair per round. Since the adversary needs to flip at least a third of the number of bits for each encoded polynomial evaluation tuple to make it bad, we have

\[
f(m) \geq m_1 + \frac{1}{3} \sum_{j=1}^{m/2} b_j
\]

\[
= m_1 + \frac{C}{3} \sum_{j=1}^{m_2/2} \left( \log \left( \frac{6Ld}{\epsilon} \right) + \left\lceil \frac{j}{d} \right\rceil \right)
\]

\[
\geq m_1 + \frac{C}{6} \left( m_2 \log \left( \frac{6Ld}{\epsilon} \right) + \frac{(m_2 - 2)^2}{4d} \right)
\]

Since the number of bits per polynomial evaluation tuple increases monotonically, the expression above becomes:

\[
f(m) \geq \begin{cases} 
    m & \text{if } m \leq d + 1 \\
    (d + 1) + \frac{C}{6} \left( (m - d - 1) \log(6Ld/\epsilon) + \frac{(m-d-3)^2}{4d} \right) & \text{otherwise}
\end{cases}
\]

Lemma 3.7. Let \( g(m) \) be the number of bits required to send \( m \) polynomial evaluation tuples, where \( m \geq d + 1 \). Then,

\[
g(m) \leq L + 5C \left( \frac{(m - d - 1)}{2} \log(6Ld/\epsilon) + \frac{(m - d + 1)^2}{8d} \right).
\]

Proof. If \( m < d + 1 \), then we have \( g(m) = m \log q \leq L \), since each of these \( m \) polynomial evaluation tuples is of length \( \log q \). For \( m > d + 1 \), taking into account the fact that each round involves exchange of at most 5 messages between Alice and Bob, we get

\[
g(m) \leq L + 5 \sum_{j=1}^{(m-d-1)/2} b_j
\]

\[
= L + 5C \sum_{j=1}^{(m-d-1)/2} \left( \log \left( \frac{6Ld}{\epsilon} \right) + \left\lceil \frac{j}{d} \right\rceil \right)
\]

\[
\leq L + 5C \left( \frac{(m - d - 1)}{2} \log(6Ld/\epsilon) + \frac{(m - d + 1)^2}{8d} \right)
\]

Lemma 3.8. Let \( L \geq 3 \), and \( r \) be any round at the end of which \( P_b \neq P_a \). Then the number of bad polynomial evaluation tuples through round \( r \) is at least \( r/4 \).

Proof. We call a field element \( x \in GF(q) \) good if \( (x, P_a(x)) \in \text{maj}(B) \), and bad otherwise. Let \( g_e \) be the number of good field elements and \( b_e \) be the number of bad field elements up to round \( r \). Similarly, let \( g_t \) be the number of good polynomial evaluation tuples and \( b_t \) be the number of bad polynomial evaluation tuples up to round \( r \). Then, from Theorem 2.2, we must have \( b_e \geq g_e - d \). Note that the total number of field elements for which Bob has received polynomial evaluation tuples from Alice through round \( r \) is \( b_e + g_e = \min(d + 2r + 1, q) \). Adding this equality to the previous inequality, we have

\[
b_e \geq \frac{1}{2} \min(2r + 1, q - d).
\]
The total number of polynomial evaluation tuples received by Bob up to round $r$ is given by

$$b_t + g_t = d + 2r + 1. \quad (3.2)$$

Note that every bad field element is associated with at least $\left\lfloor \frac{b_t + g_t}{2(b_t + g_t)} \right\rfloor$ polynomial evaluation tuples. This gives $b_t \geq b_e \left\lfloor \frac{b_t + g_t}{2(b_t + g_t)} \right\rfloor$. Using this inequality with Eqs. (3.1) and (3.2), we have

$$b_t \geq \frac{1}{2} \min(2r + 1, q - d) \left\lfloor \frac{d + 2r + 1}{2 \min(d + 2r + 1, q)} \right\rfloor \geq \frac{1}{2} \frac{d + 2r + 1}{2 \min(d + 2r + 1, q)} \min(2r + 1, q - d) \quad (3.3)$$

Case I: $(d + 2r + 1 \leq q)$ For this case, we have

$$\frac{1}{2} \left\lfloor \frac{d + 2r + 1}{2 \min(d + 2r + 1, q)} \min(2r + 1, q - d) \right\rfloor = \frac{1}{2} \frac{2r + 1}{2} \geq \frac{r}{4} \quad (3.4)$$

Case II: $(d + 2r + 1 > q)$ For this case, we have

$$\frac{1}{2} \left\lfloor \frac{d + 2r + 1}{2 \min(d + 2r + 1, q)} \min(2r + 1, q - d) \right\rfloor \geq \frac{1}{2} \frac{(d + 2r + 1)(q - d)}{2q} \geq \frac{1}{2} \frac{2r + 1}{2} \left(1 - \frac{d}{q}\right) \geq \frac{r}{4} \quad (3.5)$$

where the last inequality holds since $d/q \leq 1/3$ for $L \geq 3$.

Combining Eqs. (3.3) and (3.5), we get $b_t \geq r/4$. \qed

We now state a lemma that is crucial to the proof of Theorem 1.1.

**Lemma 3.9.** If Bob terminates before Alice, the total number of bits sent by our algorithm is

$$L + O \left( T + \log \left( T + 1, \frac{L}{\log L} \right) \log \left( \frac{L}{\epsilon} \right) \right).$$

**Proof.** Let $r'$ be the last round at the end of which $P_b \neq P_a$, or 0 if $P_b = P_a$ at the end of round 1 and for all subsequent rounds. Let $T_1$ be the number of bits corrupted by the adversary through round $r'$. Let $A_1$ represent the total cost through round $r'$ and $A_2$ be the cost of the algorithm after round $r'$. Note that after round $r'$, the adversary must corrupt one of either (1) the fingerprint, or (2) its echo, or (3) silence on the channel in Step 15 of Alice’s algorithm, in every round to delay termination. Also, after round $r'$, Alice and Bob must exchange at least a fingerprint and an echo even if $T = 0$. Thus, we have,

$$A_2 = O(T + \log(L/\epsilon)) \quad (3.6)$$

Recall that the number of polynomial evaluation tuples sent up to round $r'$ is $d + 2r' + 1$. Then, from Lemma 3.7 we have

$$A_1 \leq g(d + 2r' + 1) \leq L + 5C \left( r' \log(6Ld/\epsilon) + \frac{(r' + 1)^2}{2d} \right). \quad (3.7)$$

From Lemma 3.8, we have that the number of bad polynomial evaluation tuples is at least $\lceil r'/4 \rceil$. Thus, from Lemma 3.6 we have

$$T_1 \geq f(\lceil r'/4 \rceil) \geq \begin{cases} r'/4 \\ (d + 1) + \frac{C}{4} \left( (r'/4 - d - 1) \log(6Ld/\epsilon) + \frac{(r'/4 - d + 3)^2}{4d} \right) \end{cases} \quad \text{if } r'/4 \leq d + 1$$

if $r'/4 \leq d + 1$ otherwise. \quad (3.8)
Case I: \((r'/4 \leq d + 1)\) Since \(T_1\) is at least the number of bad polynomial evaluation tuples, from Lemma 3.8, we have \(T_1 \geq r'/4\), which gives \(r' \leq \min(4T_1, 4(d + 1))\). Hence, using Eq (3.7), we get,

\[
A_1 \leq L + 5C \left( r' \log(6Ld/\epsilon) + \frac{(r' + 1)^2}{2d} \right) \\
\leq L + 5C \left( \min(4T_1, 4(d + 1)) \log(6Ld/\epsilon) + \frac{(4d + 5)^2}{2d} \right) \\
= L + O \left( \min \left( T_1, \frac{L}{\log L} \right) \log(L/\epsilon) + \frac{L}{\log L} \right) \tag{3.9}
\]

where the last equality holds because \(d \leq L/\log L + 1\).

Case II: \((r'/4 > d + 1)\) From Eq. (3.8), we have

\[
T_1 \geq (d + 1) + C \left( \frac{(r'/4 - d - 1) \log(6Ld/\epsilon)}{4d} + \frac{(r'/4 - d + 3)^2}{4d} \right). \tag{3.10}
\]

Since each summand in the inequality above is positive and \(C > 6\), we get

\[
r' \log(6Ld/\epsilon) \leq 4T_1 + 4(d + 1) \log(6Ld/\epsilon). \tag{3.11}
\]

Since \(\frac{(r'/4 - d + 3)^2}{4d} \leq T_1\), we have \(r' \leq 8\sqrt{T_1}d + 4d - 12\). Building on this, we get,

\[
\frac{(r' + 1)^2}{2d} \leq \frac{(8\sqrt{T_1}d + 4d - 11)^2}{2d} \tag{3.12}
\]

Hence, from Eqs. (3.7), (3.11) and (3.12), we get

\[
A_1 \leq L + 5C \left( r' \log(6Ld/\epsilon) + \frac{(r' + 1)^2}{2d} \right) \\
\leq L + 5C \left( 4T_1 + 4(d + 1) \log(6Ld/\epsilon) + \frac{(8\sqrt{T_1}d + 4d - 11)^2}{2d} \right) \\
= L + O \left( T_1 + \left( \frac{L}{\log L} \right) \log(L/\epsilon) \right) \tag{3.13}
\]

where the last equality holds because \(d \leq L/\log L + 1\) and \(T_1 \geq d + 1\) from inequality (3.10).

Combining Eqs. (3.7), (3.11) and (3.13), the total number of bits sent by the algorithm becomes \(A_1 + A_2 = L + O \left( T + \min \left( T + 1, \frac{L}{\log L} \right) \log \left( \frac{L}{\epsilon} \right) \right)\).

Putting it all together, we are now ready to state our main theorem.

**Theorem 1.1.** Our algorithm tolerates an unknown number of adversarial errors, \(T\), and for a given \(\epsilon \in (0,1)\), succeeds with probability at least \(1 - \epsilon\), and sends \(L + O \left( T + \min \left( T + 1, \frac{L}{\log L} \right) \log \left( \frac{L}{\epsilon} \right) \right)\) bits.

**Proof.** By Lemmas 3.5, with probability at least \(1 - \epsilon\), Bob terminates before Alice with the correct message. If this happens, then by Lemma 3.9, the total number of bits sent is \(L + O \left( T + \min \left( T + 1, \frac{L}{\log L} \right) \log \left( \frac{L}{\epsilon} \right) \right)\). \(\square\)
4 Conclusion

We have described an algorithm for one-way interactive communication that tolerates an unknown but finite amount of noise and provides error tolerance guarantee. Against an adversary that flips $T$ bits, our algorithm succeeds with probability at least $1 - \epsilon$ and sends $L + O\left(T + \min\left(T + 1, \frac{T}{\log T}\right) \log \left(\frac{L}{\epsilon}\right)\right)$ bits, where $L$ is the length of the message with Alice, and $\epsilon \in (0, 1)$ is the error tolerance. When $\epsilon = \Omega\left(\frac{1}{\text{poly}(L)}\right)$ and $T > L$, the number of bits sent is $L + O(T)$, which is asymptotically optimal, assuming a conjecture from [14].

Several open problems remain including the following. First, can we adapt the results to interactive communication that involves more than two parties? Second, can we handle the case where $L$ is unknown to Bob? Finally, can we provide a proof to Haeupler’s conjecture, establishing the lower bounds in this setting?

References

[1] Brakerski, Z., and Kalai, Y. T. Efficient Interactive Coding against Adversarial Noise. In 53rd IEEE Annual Symposium on Foundations of Computer Science (FOCS) (2012), pp. 160–166.

[2] Brakerski, Z., and Naor, M. Fast Algorithms for Interactive Coding. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (2013), pp. 443–456.

[3] Braverman, M. Coding for Interactive Computation: Progress and Challenges. In 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (Oct 2012), pp. 1914–1921.

[4] Braverman, M. Towards Deterministic Tree Code Constructions. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS) (2012), pp. 161–167.

[5] Braverman, M., and Efremenko, K. List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise. In Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on (2014), pp. 236–245.

[6] Braverman, M., and Rao, A. Towards Coding for Maximum Errors in Interactive Communication. In Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing (STOC) (2011), pp. 159–166.

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

[8] Dani, V., Hayes, T., Movahedi, M., Saia, J., and Young, M. Interactive communication with unknown noise rate. CoRR abs/1504.06316 (2015).

[9] Dubhashi, D. P., and Panconesi, A. Concentration of measure for the analysis of randomized algorithms. Cambridge University Press, 2009.

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

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

[12] Ghaffari, M., and Haeupler, B. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding, 2013. Available at: http://arxiv.org/abs/1312.1763.
[13] Ghaffari, M., Haeupler, B., and Sudan, M. Optimal Error Rates for Interactive Coding I: Adaptivity and Other Settings. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC) (2014), pp. 794–803.

[14] Haeupler, B. Interactive channel capacity revisited. In Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on (2014), IEEE, pp. 226–235.

[15] Hashemi, M., and Trachtenberg, A. Near real-time rateless coding with a constrained feedback budget. In Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on (2014), IEEE, pp. 529–536.

[16] Luby, M. Lt codes. In null (2002), IEEE, p. 271.

[17] MacKay, D. J. Fountain codes. In Communications, IEE Proceedings- (2005), vol. 152, IET, pp. 1062–1068.

[18] Moore, C., and Schulman, L. J. Tree Codes and a Conjecture on Exponential Sums. In Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS) (2014), pp. 145–154.

[19] Naor, J., and Naor, M. Small-bias probability spaces: Efficient constructions and applications. SIAM journal on computing 22, 4 (1993), 838–856.

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

[21] Palanki, R., and Yedidia, J. S. Rateless codes on noisy channels. In IEEE International Symposium on Information Theory (2004), Citeseer, pp. 37–37.

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

[23] Reed, I. S., and Solomon, G. Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics 8, 2 (1960), 300–304.

[24] Schulman, L. Communication on Noisy Channels: A Coding Theorem for Computation. In Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on (Oct 1992), pp. 724–733.

[25] Schulman, L. J. Deterministic Coding for Interactive Communication. In Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC) (1993), pp. 747–756.

[26] Welch, L. R., and Berlekamp, E. R. Error correction for algebraic block codes, Dec. 30 1986. US Patent 4,633,470.