Interactive Coding for Interactive Proofs

Allison Bishop*
Columbia University
alewko@cs.columbia.edu

Yevgeniy Dodis†
New York University
dodis@cs.nyu.edu

Abstract

We consider interactive proof systems over adversarial communication channels. We show that the seminal result that **IP** = **PSPACE** still holds when the communication channel is malicious, allowing even a constant fraction of the communication to be arbitrarily corrupted.

*Supported in part by NSF CNS 1413971 and NSF CCF 1423306.
†Partially supported by gifts from VMware Labs and Google, and NSF grants 1319051, 1314568, 1065288, 1017471.

1 Introduction

Interactive proofs are fundamental objects in both cryptography and complexity theory, and come with a rich history of exciting developments, such as the surprising characterization that **IP** = **PSPACE** [25]. This characterization assumes that a prover and a verifier communicate over a perfect communication channel, and crucially relies upon the fact that the number of rounds of the interaction can be polynomially long.

Recently, the study of interactive coding (pioneered by Schulman [22, 23]) has emerged as a promising way to extend results involving lengthy interactions over perfect channels to analogous results over adversarial channels - even with a constant relative error rate. This high level of robustness cannot be achieved by simply applying an error correcting code to each message, a method which is limited to an error rate proportional to $\frac{1}{r}$, where r is the number of rounds. There has been much success in obtaining interactive coding protocols capable of performing any two party communication tasks over a noisy or adversarial channel [22, 23, 7, 11, 5, 4, 19, 1, 14, 13, 6, 17]. However, all of these works assume that the task is described as a function of two inputs, and only correctness of the computation is required.

In the case of interactive proofs, it is not enough to ensure that an honest party "eventually" learns the real message the other party was attempting to send. Instead, we must ensure that the interference of the channel cannot prevent an honest prover from convincing a verifier of a true statement, and also cannot help a malicious prover convince a verifier of a false statement. This appears to be problematic if we consider the techniques employed by interactive coding protocols, which enable parties to "replay" and "revise" their messages as the interactive coding mechanism runs. We must worry, then, that a malicious prover may use the excuse of potential channel errors to change its responses adaptively after peeking ahead at the verifier's future challenges. For this reason, it does not suffice to simply take an interactive proof system designed for an error-free channel and compile it blindly using an off-the-shelf interactive coding method.

An undaunted optimist might then ask for strong interactive coding mechanism, one that could provably compose with a wide variety of security properties, such as soundness for interactive proof systems or input privacy for multiparty computation. The most general version of this would achieve a notion ensuring that the participants in the error-resilient version of the protocol "do not learn" anything more than they would learn from executing the error-free protocol. A formalization of this called "knowledge-preserving interactive coding" was introduced and studied by Chung, Pass, and Telang [9], who showed that under this strong requirement, applying an error-correcting code in each round is essentially optimal. This means no error rate beyond $\frac{1}{r}$ is possible without making computational assumptions. Similarly, the work of Gelles, Sahai, and Wadia [12] proves an impossibility result for error-resilient secure multiparty computation.

In contrast, we show that *for interactive proofs a constant relative error rate can be achieved.* In other words, while positive results in traditional interactive coding only (successfully) handle *correctness*, and negative results rule out (strong forms of) *zero-knowledge/privacy*, we show that ensuring *soundness* is still feasible in the presence of adversarial communication noise. This is perhaps a bit counterintuitive, as the negative results proceed by proving that some amount of backtracking and replaying messages is inherent for this level of error-correction. Nonetheless, using amplification techniques, we can preprocess our interactive proofs to withstand a certain amount of backtracking, since the verifier is aware of the backtracking, and can sample fresh randomness to mitigate the potential advantage gained by a malicious prover as a consequence.

One additional challenge we face is that the verifier must remain efficient, meaning that the encoding and decoding for the interactive coding mechanism must be computable in polynomial time. Many of the interactive coding results are existential rather than efficient (e.g. [22, 23, 7]), but the recent work of Brakerski and Kalai [4] managed to obtain computationally efficient interactive coding protocols, even for constant rate adversarial errors. We employ a simplified version of their techniques, obtaining our simplifications due to the fact that the individual messages of our protocols can be taken to be not too short. This allows us to avoid the use of expensive "tree codes" that are needed in [4], and results in much easier to understand protocols.

Our Techniques. It is well-known how to design an interactive proof system for **PSPACE** that has both perfect completeness (an honest prover can convince a verifier of a true statement with probability 1) and very small soundness error (a malicious prover can only convince a verifier of a false statement with very small probability). Starting from such a system, we observe that even if a malicious prover could make a verifier re-sample a particular challenge polynomially many times, the chance of obtaining a value that would allow "cheating" remains reasonably small. We can thus hope to withstand a certain fraction of channel errors by allowing the protocol to "backtrack" when errors occur, while having the verifier resample its randomness to control the potential gain for a malicious prover. Our coding techniques will prevent a malicious prover from changing its answer to a previous challenge, instead requiring it to answer a new challenge if it uses the potential errors as an excuse to backtrack.

We organize our proof into two separate tasks: first obtaining an **IP** system that still works over perfect channels, but allows parties to arbitrarily signal that they want to back up one round of interaction at a time. We call such a system "backtracking-resilient." We then design a compilation procedure that takes any backtracking-resilient **IP** system and produces a new proof system that works over adversarial channels, allowing a constant error rate. This compilation relies upon hashing techniques that are reminiscent of the efficient compiler in [4]. In particular, we have both parties hash their current simulated transcripts with freshly chosen keys at each exchange, so that they can detect any disagreements whenever the channel does not introduce too many errors.

There are two noteworthy features of our compiler. First, since the backtracking-resilient **IP** system we use anyway has reasonably large individual messages, we can avoid the use of expensive tree codes that are needed in [4] to protect the "simulation units between the hash stages". Second, we give an explicit reduction between the soundness of our compiled protocol and the notion of backtracking resilience. This formalizes the intuition that the hashing techniques essentially limit a malicious prover impersonating an adversarial channel to choosing when to backtrack the protocol, and could be useful for future work.

2 Resilient Interactive Protocols

Interactive Protocols. We recall the notion of an r-round interactive protocol Π between a deterministic prover $\mathcal{P}$ and a probabilistic verifier $\mathcal{V}$ (who share some common input x which we omit, when clear). We view the verifier $\mathcal{V}$ as an algorithm that takes in a partial transcript and some fresh randomness and outputs a next message.[1] More formally, $\mathcal{V} : \mathcal{T} \times Rand \rightarrow \{0,1\}^*$, where $\mathcal{T}$ is the set of partial transcripts and $Rand$ is the set of random values. We view the prover

[1]This view is without loss of generality, since it is known that private-coin protocols can be simulated by public-coin ones [16], meaning that $\mathcal{V}$ never needs to keep any state beyond its partial transcript so far.

$\mathcal{P}$ as a deterministic algorithm that takes in a partial transcript and outputs a next message: $\mathcal{P} : \mathcal{T} \to \{0,1\}^*$.

In the typical (error-free) setting, the protocol proceeds in some number of rounds, r, where in each round i the verifier sends a challenge C_i and the prover sends a response R_i. If we let τ_{i-1} denote the transcript after $i-1$ rounds, the i^{th} round consists of the verifier sampling a fresh random value $c_i \in Rand$ and sending the challenge $C_i := \mathcal{V}(\tau_{i-1}, c_i)$. The prover then sends the response $R_i := \mathcal{P}(\tau_{i-1}||C_i)$. We then have $\tau_i = \tau_{i-1}||C_i||R_i$.

We assume each party locally stores its own copy of the partial transcript. We let τ_p denote the prover's (evolving) copy and τ_v denote the verifier's (evolving) copy of the transcript. (Note, in the error-free settings these values are always consistent; however, once we are allowing channel errors, these two partial transcripts may temporarily diverge.) At the end of round r, $\mathcal{V}$ either accepts or rejects the final transcript τ_v, and we denote the random variable (over the coins of $\mathcal{V}$) indicating this decision by $(\mathcal{P}, \mathcal{V})$.

Given a language L, we say that Π is *(perfectly) complete* on L, if for any $x \in L$, we have $\Pr[(\mathcal{V}(x), \mathcal{P}(x)) \to accept] = 1$. Similarly, Π is ε-sound on L, if for any $x \notin L$ and any potentially cheating prover $\tilde{\mathcal{P}}$, we have $\Pr[(\mathcal{V}(x), \tilde{\mathcal{P}}(x)) \to accept] \leq \varepsilon$.

Definition 1. *We say that L belongs to the class* **IP** *(Interactive Protocols), if there exist polynomial $r = r(n)$ and $t = t(n)$ and an r-round interactive protocol $(\mathcal{P}, \mathcal{V})$ where the running time of $\mathcal{V}$ on n-bit inputs x is at most $t(n)$ and: (a) Π is (perfectly) complete; (b) Π is $(1/2)$-sound.*

Of course, repeating Π in parallel λ times, we can reduce the soundness error to $\varepsilon = 2^{-\lambda}$, for any polynomial $\lambda = \lambda(n)$. It is known [25] that **IP** = **PSPACE**, the class of languages decided with polynomial space.

Error-Resilient Protocols. To define error-resilient protocols, we must specify the power of an adversarial channel. We will model an adversarial channel as an algorithm $\mathcal{A}$ that intercepts messages as they are sent and may modify them arbitrarily in transit. We make no restrictions on the computational power of $\mathcal{A}$ (it may be unbounded) and also allow it to know the entire state of the prover, the verifier, and the partial transcript at any point during the execution. It does not know, however, the future randomness to be selected by the verifier.

Definition 2. *(*Completeness with Adversarial Channel Error*) Given a language L, we say that a T-round protocol Π is (α, δ)-error-complete on L, if the following condition holds for any $x \in L$. For any (unbounded) adversary $\mathcal{A}$ that can cause at most an α-fraction of errors throughout the entire communication, the honest prover $\mathcal{P}_\alpha$ will convince the honest verifier $\mathcal{V}_\alpha$ with probability $> \delta$.*

For soundness in the presence of adversarial noise, we observe that we can always "merge" the adversarial prover $\tilde{\mathcal{P}}$ with our channel adversary $\mathcal{A}$, simply resulting in a different adversarial prover $\tilde{\mathcal{P}}'$. In other words, soundness with adversarial channel error is equivalent to traditional soundness!

Definition 3. *We say that L belongs to the class* **ERIP** *(Error-Resilient* **IP***), if there exists a constant $\alpha > 0$, polynomials $T = T(n)$ and $t = t(n)$, and a T-round interactive protocol $\Pi = (\mathcal{P}_\alpha, \mathcal{V}_\alpha)$ where the running time of $\mathcal{V}_\alpha$ on n-bit inputs x is at most $t(n)$ and: (a) Π is $(\alpha, \frac{2}{3})$-error-complete; (b) Π is $(1/3)$-sound.*

Just like for (imperfect completeness) **IP**, the completeness and soundness constants 2/3 and 1/3 of **ERIP** can be amplified to $(1-2^{-\Omega(p)})$ and $2^{-\Omega(p)}$, respectively, by doing p parallel repetitions $\Pi_1, \ldots, \Pi_p$, and taking the majority vote. A small subtlety in this (otherwise immediate) argument comes from analyzing completeness in the presence of errors (soundness is the same as for **IP**). This is because the attacker $\mathcal{A}$ can split his errors non-uniformly across the p repetitions Π_i, causing failures with high probability for repetitions where more than α-fraction of errors were introduced. Fortunately, by setting the robustness threshold α' of the new parallel protocol Π^* to be $\alpha' = \alpha/10$, we can apply Markov's inequality to conclude that $\mathcal{A}$ can cause more than α-fraction of errors on at most $p/10$ sub-protocols Π_i, meaning that $\mathcal{A}$ would still need to break $(\alpha, 2/3)$-error-completeness for at least $p/2 - p/10 = 2p/5$ protocols Π_i in order to break the error-completeness of Π^*. However, since in any of the p protocols $\mathcal{A}$'s success of doing so is at most 1/3, the honest verifier acts independently across the p runs, and $2p/5 > p/3$, we can use the Chernoff bound to conclude that $\mathcal{A}$'s overall success probability will be $2^{-\Omega(p)}$.

Main Result. Our main result can then be stated as:

Theorem 4. (Main Result) ERIP = IP = PSPACE.

We will prove this over the course of the next three sections.

3 Backtracking-Resilient Protocols

As an intermediary step in achieving error-resilient proof systems, we will define proof systems that retain their completeness and soundness properties under a milder disruption we call "backtracking." In other words, we augment usual error-free protocols with an additional mechanism for backtracking. In addition to sending a challenge or response, either party may at any time transmit a special symbol B instead. Upon sending or receiving a B, the parties each remove the latest complete round from their partial transcripts. For example, suppose $\tau_p = \tau_v = \tau_i$ at the time a B is sent/received. Then both parties revert to τ_{i-1} and will start again with the verifier choosing *fresh randomness* to send a (potentially) new challenge to $\mathcal{P}$ for round i (it is as if the old version of round i never happened).

We let U be the maximal number of backtracking steps (i.e., the "budget") allowed by each party. Given any standard interactive protocol Π and any such budget U, we obtain U-backtracking extension of Π, Π_U, where the modified prover $\mathcal{P}_U$ (resp. verifier $\mathcal{V}_U$) is identical to the honest prover $\mathcal{P}$ (resp. verifier $\mathcal{V}$), except allowing allow up to U backtracking steps to the communicating partner, as described above. In particular, $\mathcal{V}_U$ will output the same decision as $\mathcal{V}$ when the transcript τ_v reaches the last round (for the first time), but also $\mathcal{V}_U$ will reject if more than U backtracking steps are attempted by the (possibly malicious) prover. Thus, if Π has r rounds, without loss of generality we can cap the number of rounds of Π_U by $T = r + 4U$, as each of the (at most $2U$) backtracking steps requires one extra "normal step" to get back, meaning that in at most $r + 4U$ rounds the transcript τ_v is guaranteed to reach the decision point for $\mathcal{V}$.

Of course, when playing against themselves, $\mathcal{P}_U$ and $\mathcal{V}_U$ will not use back backtracking steps, and the protocol will terminate in r rounds. However, we would like to extend completeness of soundness condition to hold even if (possibly malicious) backtracking is allowed. For the former, we will assume that $\mathcal{P}_U$ and $\mathcal{V}_U$ are honest, except for the adversarial backtracking steps (see below);

for the latter, we will assume that the verifier $\mathcal{V}_U$ is honest, but the prover $\tilde{\mathcal{P}}_U$ is malicious, including (wlog, up to U) backtracking steps. This is formalized below.

Definition 5. *(*Perfect Completeness with U-Backtracking*) Given a language L, we say that an r-round protocol* Π *is* (perfectly) U-backtracking-complete *on L, if the following condition holds with probability* 1*, for any $x \in L$. Let $T = r + 4U$ and $C_p, C_v \in \{\perp, B\}^T$ be two strings of length T containing at most U occurrences of the symbol B. We let $C_p(i)$ denote the i^{th} symbol of C_p, and same for $C_v(i)$. We require that if an honest prover $\mathcal{P}_U$ and verifier $\mathcal{V}_U$ run the protocol for a true statement, except with the prover sending B in round i whenever $C_p(i) = B$ and the verifier sending B in round i whenever $C_v(i) = B$, then the verifier accepts after at most T rounds with probability* 1*.*

Definition 6. *(*Soundness with U-Backtracking*) Given a language L, we say that an r-round protocol* Π *is* (U, ε)-backtracking-sound *on L, if for $x \notin L$, and any malicious prover $\tilde{\mathcal{P}}_U$ for the U-backtracking extension Π_U of* Π*,* $\Pr[(\mathcal{V}_U(x), \tilde{\mathcal{P}}_U(x)) \rightarrow accept] \leq \varepsilon$*.*

Lemma 7. *Assume* Π *is an r-round interactive protocol which is complete and ε-sound for some langueage L. Then, for any U,* Π *is U-backtracking-complete and (U, ε')-backtracking-sound, where*

$$\varepsilon' \leq \varepsilon \cdot 2^{r+4U}$$

Proof. Every final transcript produced with non-zero probability in Π_U must also occur with non-zero probability without backtracking, using the original algorithms. Thus, perfect completeness for the underlying algorithms implies perfect completeness with backtracking.

To prove soundness, we fix a false statement x and an arbitrary malicious prover $\tilde{\mathcal{P}}_U$ for Π_U. Let ε' be the probability that $\tilde{\mathcal{P}}_U(x)$ convinces $\mathcal{V}_U(x)$. Given any *fixed* sequence $C \in \{\perp, B\}^{r+4U}$ of possible backtracking steps of $\tilde{\mathcal{P}}_U$ containing at most U occurrences of the symbol B, we let ε'_C be the probability that $\tilde{\mathcal{P}}_U(x)$ succeeds and *precisely* respects the backtracking sequence C. Clearly, since all such events are disjoint, we have $\varepsilon' = \sum_C \varepsilon'_C$. To complete the proof, it suffices to show that $\varepsilon'_C \leq \varepsilon$, for any fixed C, as the number of C's is at most 2^{r+4U}.

To show that $\varepsilon'_C \leq \varepsilon$, we define a malicious prover $\tilde{\mathcal{P}}^C$ of the original (non-backtracking) protocol Π whose success probability is at precisely ε'_C, which implies that $\varepsilon'_C \leq \varepsilon$, by standard ε-soundness. Given C, $\tilde{\mathcal{P}}^C$ can pre-compute all the r rounds $1 \leq i_1, \ldots, i_r \leq r + 4U$ which will not be "erased" from the final transcript of $\mathcal{V}_U$ *assuming that $\tilde{\mathcal{P}}_U$ follows C*. Then $\tilde{\mathcal{P}}^C(x)$ emulates $\tilde{\mathcal{P}}_U(x)$ as follows: (a) the challenges for all the "non-erased" rounds $i_1, \ldots, i_r$ are obtained from the honest verifier $\mathcal{V}$; (b) the challenges from the remaining "erased" rounds are honestly generated by $\tilde{\mathcal{P}}^C$ himself; (c) if at any point $\tilde{\mathcal{P}}_U$ generates a backtracking step inconsistent with C, $\tilde{\mathcal{P}}^C$ aborts. Since the above emulation is identical to the real run of $\tilde{\mathcal{P}}_U$ when consistent with C, $\tilde{\mathcal{P}}^C$ succeeds with probability $\varepsilon'_C \leq \varepsilon$, as claimed. □

Definition 8. *We say that L belongs to the class* **BRIP** *(Backtracking-Resilient* **IP***), if there exist polynomial $r = r(n)$ such that for any polynomial $U = U(n)$ there exists a polynomial $t = t(n)$ and an r-round interactive protocol* $\Pi = (\mathcal{P}, \mathcal{V})$ *where the running time of $\mathcal{V}$ on n-bit inputs x is at most $t(n)$ and: (a)* Π *is (perfectly) U-backtracking-complete; (b)* Π *is* $(U, 1/2)$*-backtracking-sound.*

Using Lemma 7, we observe that the class of interactive protocols is backtracking-resilient.

Corollary 9. **BRIP** = **IP** = **PSPACE**.

Proof. Take any $L \in \mathbf{IP}$. This means L has an r-round, $1/2$-sound interactive protocol Π for some polynomial r. Now take any polynomial U for the backtracking budget. By repeating Π in parallel $r + 4U + 1$ times, we get protocol Π' for L which still has r rounds, polynomial-time verifier, is complete and ε-sound, where $\varepsilon = 2^{-r-4U-1}$. By Lemma 7, Π' is U-backtracking-complete and $(U, 1/2)$-backtracking-sound, completing the proof. $\square$

Remark 10. *We can easily reduce the soundness error* $1/2$ *to be exponentially small, either by directly adjusting the proof of Corollary 9, or by doing parallel repetition on any* **BRIP** *protocol.*

4 Compiling Backtracking-Resilient Protocols Against Adversarial Channel Errors

We now present a method for taking a backtracking-resilient interactive proof system and compiling it into one that can resist a constant rate of adversarial channel errors. Intuitively, the prover and verifier will attempt to simulate the backtracking-resilient protocol over the adversarial channel. They will use hash functions with freshly chosen keys each time to check if they are in agreement on the partial transcript simulated so far. Every message and hash key will be encoded with an error correcting code, to ensure that the adversary must invest a high amount of errors to cause confusion between the parties. Of course, sometimes channel errors will still prevent the parties from detecting an inconsistency in the simulated transcript. But the adversary cannot afford to keep up this high error investment indefinitely, and eventually the parties will detect the problem and backtrack to fix it. This will result in a simulated transcript that mimics an execution of the backtracking-resilient protocol, and hence appropriate analogs of completeness and soundness for this compiled protocol can be reduced to backtracking-resilience of the underlying protocol.

We prove the following result, which, by Corollary 9, suffices to establish our main result in Theorem 4.

Theorem 11. $\mathbf{ERIP} = \mathbf{BRIP}$.

Since $\mathbf{ERIP} \subseteq \mathbf{IP} = \mathbf{BRIP}$, we only need to show that $\mathbf{BRIP} \subseteq \mathbf{ERIP}$. Before proving this result, we need some standard tools from hashing and coding.

Hashing and Coding. We will use a family of hash functions indexed by keys $k \in \{0,1\}^\gamma$. More precisely, we invoke the following theorem also used in [4]:

Theorem 12. *([20, 2]). There exists a constant* $q > 0$ *and an ensemble of hash families* $\{H_N\}_{N \in \mathbb{N}}$ *such that for every* $N \in \mathbb{N}$ *and for every* $h \in H_N$, $h : \{0,1\}^{\leq 2^N} \to \{0,1\}^{qN}$ *is poly-time computable, it is efficient to sample* $h \leftarrow H_N$ *using only* qN *random bits, and for all* $y \neq z \in \{0,1\}^{\leq 2^N}$ *it holds that*

$$\Pr_{h \to H_N}[h(y) = h(z)] \leq 2^{-N}.$$

We let $\gamma = qN = O(N)$ and write $h_k : \{0,1\}^{\leq 2^N} \to \{0,1\}^\gamma$ to denote the element of H_N sampled with the random string $k \in \{0,1\}^\gamma$. We also let *Encode* and *Decode* denote the encoding and decoding algorithms of an error-correcting code with a constant rate and a constant relative distance β.

Our Compiler. Take any $L \in \mathbf{BRIP}$ which means that L has an r-round backtracking-resilient protocol Π, for any polynomially bounded budget U. (As we will see shortly, we will only use $U = O(r)$.) To show $L \in \mathbf{ERIP}$, we set $\alpha = \Omega(\beta)$ to be the constant error rate we will tolerate on the channel, and show how to build an error-resilient proof system for L tolerating an α-fraction of adversarial errors. Our new protocol $\tilde{\Pi}$ will run for $\widetilde{T}$ rounds, where we define $\widetilde{T}$ such that $\widetilde{T}(1 - 18\alpha\beta^{-1}) = r$. In particular, when α is chosen to be a suitably small constant fraction of β (e.g., $\alpha = \beta/36$), this is possible with $\widetilde{T} = O(r)$ (e.g., $\tilde{T} = 2r$).

We define the backtracking budget U of our original protocol $\Pi = (\mathcal{P}, \mathcal{V})$ by $U = 9\alpha\beta^{-1}\widetilde{T} = O(r)$ (e.g., for $\alpha = \beta/36$, we have $U = r/2$), and assume that Π is $(U, \frac{1}{3})$-backtracking sound and perfectly U-backtracking complete (guaranteed possible by Lemma 7). We also denote by $T = r + 4U = O(r)$ (e.g., $T = 3r$ when $\alpha = \beta/36$) the maximal number of rounds of the U-backtracking extension Π_U of Π, by ℓ the length of the challenges to be sent by the verifier $\mathcal{V}$ in each round, and assume that the hashing parameters $\gamma, N = \Omega(\log r)$. Finally, we will use *Encode* and *Decode* for encoding/decoding messages of length $\ell + 2\gamma + \log(T)$.

We can now describe the new algorithms $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{V}}$ for an interactive proof system that can resist adversarial channel errors at a constant rate α. These algorithms will run for $\widetilde{T}$ message exchanges, where each exchange will still consist of a message sent by the verifier and then a response sent by the prover, and will require only black-box access to $\mathcal{P}$ and $\mathcal{V}$.

$\widetilde{\mathcal{P}}$ will maintain internal variables $\tilde{\tau}_p$ and $\tilde{i}_p$. These will function as the prover's internal views of the simulated transcript and round number respectively. $\widetilde{\mathcal{V}}$ will similarly maintain internal variables $\tilde{\tau}_v$, $\tilde{i}_v$, and $\tilde{C}$. These will function as the verifier's internal views of the simulated transcript, the round number, and the pending challenge. We initialize $\tilde{\tau}_p$, $\tilde{\tau}_v$ and $\tilde{C}$ to $\emptyset$, which denotes the empty string. We initialize $\tilde{i}_p$ and $\tilde{i}_v$ to 1.

The First Round: $\widetilde{\mathcal{V}}$ will start a run of $\mathcal{V}$ to obtain a first challenge C. It sets $\tilde{C} = C$. It also samples a uniformly random hash key $k_1 \in \{0,1\}^\gamma$. It will send to the prover: $\mathit{Encode}(C||k_1||h_{k_1}(\tilde{\tau}_v)||\tilde{i}_v)$.

The Prover's Algorithm: In any round, when the prover receives a message from the verifier, it decodes it as a challenge C, a key k, a hash value h, and a round index i. It then performs the following steps to update its internal variables and produce a response:

- If $h_k(\tilde{\tau}_p) \neq h$ and $i \leq \tilde{i}_p$, then decrement $\tilde{i}_p$, erase a round from $\tilde{\tau}_p$, and set R equal to the last prover response now reflected in $\tilde{\tau}_p$.
- If $h_k(\tilde{\tau}_p) \neq h$ and $i > \tilde{i}_p$, then keep $\tilde{i}_p, \tilde{\tau}_p$ the same, and set R equal to the last prover response reflected in $\tilde{\tau}_p$.
- If $h_k(\tilde{\tau}_p) = h$, then set $R = \mathcal{P}(\tilde{\tau}_p||C)$, concatenate $C||R$ onto $\tilde{\tau}_p$, and increment $\tilde{i}_p$.

The prover then chooses a new uniformly random key k' and sends $\mathit{Encode}(R||k'||h_{k'}(\tilde{\tau}_p)||\tilde{i}_p)$.

The Verifier's Algorithm: When the verifier receives a message from the prover, it decodes it as a response R, a key k, a hash value h, and a round index i. It then performs the following steps to update its internal variables and produce a response:

- If $h_k(\tilde{\tau}_v||\tilde{C}||R) \neq h$ and $i \leq \tilde{i}_v$, then decrement $\tilde{i}_v$, erase a round from $\tilde{\tau}_v$, and set $\tilde{C} = \mathcal{V}(\tilde{\tau}_v, \mathit{rand})$, for a freshly chosen random value rand.

- If $h_k(\tilde{\tau}_v||\tilde{C}||R) \neq h$ and $i > \tilde{i}_v$, then keep $\tilde{i}_v, \tilde{\tau}_v, \tilde{C}$ the same.
- If $h_k(\tilde{\tau}_v||\tilde{C}||R) = h$, then concatenate $\tilde{C}||R$ onto $\tilde{\tau}_v$, increment $\tilde{i}_v$, and then set $\tilde{C} = \mathcal{V}(\tilde{\tau}_v, rand)$ for a freshly chosen random value $rand$.

The verifier then chooses a new uniformly random key k' and sends $Encode(\tilde{C}||k'||h_{k'}(\tilde{\tau}_v)||\tilde{i}_v)$.

At the end of the $\widetilde{T}$ message exchanges, the verifier outputs the decision of $\mathcal{V}(\tilde{\tau}_v)$.

Efficiency. Using any efficient hashing and coding scheme, our new prover and verifier algorithms are efficient given oracle access to the (next message function) of the original prover and verifier.

5 Analysis of the Compiled Algorithms

We now prove that the algorithms $\widetilde{P}, \widetilde{V}$ presented in the previous section satisfy completeness and soundness despite adversarial channel error.

5.1 Completeness

We first seek to prove completeness. For this, we assume an honest prover and an adversarial channel that can cause at most an α-fraction of errors throughout the entire communication. We prove:

Lemma 13. *If $\mathcal{P}, \mathcal{V}$ is perfectly U-backtracking complete, then $\widetilde{\mathcal{P}}, \widetilde{\mathcal{V}}$ is $(\alpha, \frac{2}{3})$-error-complete.*

Proof. We will define a measure of progress, M, that will potentially oscillate as the protocol runs. At the beginning of any particular exchange (just before the verifier sends its next message), we can determine the value of M as follows. First, we let m be the maximal number of rounds such that $\tilde{\tau}_p$ and $\tilde{\tau}_v$ agree on a prefix of m rounds. We then set

$$M := m - (\tilde{i}_p - m) - (\tilde{i}_v - m).$$

We define a *good exchange* as follows. First, we require that the verifier has correctly decoded the previous message sent by the prover. Additionally, we require that the two messages sent during the exchange (by the verifier and then by the prover) are also decoded correctly. We refer to any other exchange as a *bad exchange.*

Lemma 14. *A bad exchange decreases M by at most 3.*

Proof. In any exchange, the value of m can decrease by at most 1, since at most one round of the simulated transcripts is erased at a time. Since each of $\tilde{i}_p, \tilde{i}_v$ can be incremented by at most 1 in any exchange, we then have that the total decrement in M is bounded by 3. $\square$

Lemma 15. *Conditioned on the event that there are no hash collisions, a good exchange increases M by at least 1.*

Proof. Suppose at the beginning of a good exchange, the verifier has calculated that the current hash value agrees, and has just incremented $\tilde{i}_v$. Since we are assuming no hash collisions have occurred, this implies that $\tilde{\tau}_v = \tilde{\tau}_p$ and $\tilde{i}_v = \tilde{i}_p$ at this point. $\mathcal{V}$ will choose a new challenge $\tilde{C}$, send this to $\mathcal{P}$, who will form a response R, and both $\mathcal{V}$, $\mathcal{P}$ will concatenate $\tilde{C}||R$ onto their transcripts.

In this case, m will increase by 1, and $\tilde{i}_p - m, \tilde{i}_v - m$ will both remain 0. Hence M will increase by 1.

Now suppose instead that at the beginning of a good exchange, the verifier has detected a disagreement in the hash values. In the case that $\tilde{i}_v \geq \tilde{i}_p$, the verifier will erase a round from $\tilde{\tau}_v$ and decrement $\tilde{i}_v$. This will lead to a decrease in $\tilde{i}_v - m$ but no decrease in m. When $\mathcal{P}$ correctly decodes the next message from $\mathcal{V}$, a decrease in m is impossible since an agreed upon round cannot be erased when $\tilde{i}_p = \tilde{i}_v$ (since the hash values will agree in this case) and when $\tilde{i}_p$ remains less than $\tilde{i}_v$, the prover will not erase a round from $\tilde{\tau}_p$. Also an increase in $\tilde{i}_p - m$ is impossible, since $\tilde{i}_p$ will only be incremented if a new agreed upon round is being added to the simulated transcript. Thus M will also increase in this case.

We next consider the case where the verifier has detected a disagreement in the hash values and $\tilde{i}_v < \tilde{i}_p$. The verifier will then leave $\tilde{\tau}_v, \tilde{i}_v$ unchanged, but the prover will decrement $\tilde{i}_p$ and erase a round from $\tilde{\tau}_p$, leading to a decrease in $\tilde{i}_p - m$, and hence an increase in M.

□

We now observe that a bad exchange can be extended to a *bad interval* containing three transmitted encodings - the previous message from the prover to the verifier, and the two messages in the bad exchange itself. At least one of these messages must have been corrupted beyond its capacity, resulting in a relative error rate within this interval of $> \frac{\beta}{3}$.

We note the following lemma stated in [24]:

Lemma 16. *(Lemma 7 in [24]) In any finite set of intervals on the real line whose union is of total length s, there is a subset of disjoint intervals whose union is of total length at least $\frac{s}{2}$.*

We suppose there are s bad exchanges during a run of $\widetilde{T}$ total exchanges. Then there are at least $\frac{s}{2}$ bad exchanges whose corresponding bad intervals are disjoint. This results in a total relative error rate of $\frac{s\beta}{6\widetilde{T}}$. We must have: $s \leq 6\alpha\beta^{-1}\widetilde{T}$.

Thus, after $\widetilde{T}$ exchanges if no hash collisions have occurred, the value of our progress measure M satisfies $M \geq \widetilde{T} - 18\alpha\beta^{-1}\widetilde{T}$, which we can rewrite as $M \geq \widetilde{T}(1 - 18\alpha\beta^{-1})$. Recall that this is $\geq r$ by our choice of $\widetilde{T}$. This implies that the simulated transcript will be a full r rounds of a transcript that occurs with non-zero probability in the backtracking resilient algorithms over clear channels. We also observe that if $\widetilde{U}$ is the number of backtracks occurring during a particular execution, then $\widetilde{T} - 2\widetilde{U} \geq M$, so because we set $U := 9\alpha\beta^{-1}\widetilde{T}$, we have ensured that at most U backtracks occur. Thus, completeness follows from the perfect completeness of the underlying backtracking resilient algorithms if we choose parameters that make the probability of a hash collision $< \frac{1}{3}$.

To bound the probability of hash collisions, we employ Theorem 12 and a union bound to conclude that the probability of a hash collision occurring at any time throughout the protocol simulation is $O(\widetilde{T}2^{-N}) = O(r2^{-N})$. Thus it suffices to set N proportional to $\log(r)$ to achieve a bound $< \frac{1}{3}$.

Putting this all together, we have proven Lemma 13. □

5.2 Soundness

Next, we show that soundness is preserved by our compiler, irrespective of the value U.

Lemma 17. *If $\mathcal{P}, \mathcal{V}$ is $(U, \frac{1}{3})$-backtracking-sound, then $\widetilde{\mathcal{P}}, \widetilde{\mathcal{V}}$ is $\left(\frac{1}{3}\right)$-sound.*

Proof. We are considering a perfect channel and a malicious prover who seeks to convince the verifier of a false statement. The verifier, of course, does not know the errors are not coming from the channel. We fix a false statement and a malicious prover $\widetilde{\mathcal{P}}$ who manages to convince $\widetilde{\mathcal{V}}$ to accept with probability $> \frac{1}{3}$. From this, we will create a malicious prover $\mathcal{P}$ for the underlying back-tracking resilient algorithm that contradicts soundness with backtracking.

The Malicious $\mathcal{P}$: The malicious prover $\mathcal{P}$ for the backtracking-resilient proof system behaves as follows. It will run the malicious prover $\widetilde{\mathcal{P}}$ internally, simulating the messages from $\mathcal{V}$. It initializes $\widetilde{\mathcal{P}}$ with the same false statement to be proved, and initializes internal variables $\tilde{\tau}_v, \tilde{i}_v, \tilde{C}$.

When $\mathcal{V}$ submits a challenge C, $\mathcal{P}$ chooses a random hash key k and sends $Encode(C||k||H_k(\tilde{\tau}_v)||\tilde{i}_v)$. It updates $\tilde{C} = C$. Upon receiving a response from $\widetilde{\mathcal{P}}$, it decodes it and parses the result as a tuple (R, k, h, i). It then internally performs the algorithm of $\widetilde{\mathcal{V}}$. If the result is a decrement to $\tilde{i}_v$ and the erasure of round from $\tilde{\tau}_v$, then $\mathcal{P}$ sends B to $\mathcal{V}$. If the result is an increment to $\tilde{i}_v$, it sends R to $\mathcal{V}$. If the result is no change, it simulates the next message to $\widetilde{\mathcal{P}}$. It continues simulating $\widetilde{\mathcal{V}}$ in this way.

By construction, $\widetilde{V}$ will accept in this simulation only when $\mathcal{V}$ accepts. Hence this malicious prover $\mathcal{P}$ can falsely convince $\mathcal{V}$ with probability $> \frac{1}{3}$. □

We observe that Lemmas 13 and 17 imply Theorem 11. Taken together, Theorem 11 and Corollary 9 imply Theorem 4.

6 Conclusions and Open Problems

We showed the feasibility of interactive coding for interactive protocols, tolerating a constant fraction of adversarial communication errors. Additionally, our compiled error-resilient protocol is within a constant factor from optimal in its round complexity, and has an honest prover/verifier which is efficient given oracle access to the original prover/verifier. We also believe that our result should "scale down" to the setting of "interactive proofs for muggles" considered by Goldwasser, Kalai and Rothblum [15], who showed how to achieve polynomial-time, communication-efficient interactive proofs with $O(n \cdot polylog(n))$ verification time for any language in **NC** (class of uniform, polynomial size and polylogarithmic depth circuits).

We now list several interesting open problems for future work.

Better Communication Complexity. Unlike its asymptotically optimal round complexity and error rate, our compiler incurs an $O(r)$ overhead in communication complexity, as compared to the error-free setting (where r is the number of rounds in the error-free setting). This is due to the $O(r)$ parallel repetition used to amplify the soundness of the original protocol Π in Corollary 9. We make several remarks about this blowup.

First, we can use a more randomness efficient parallel repetition for public-coin **IP** due to Bellare et al. [3]. This will reduce the communication complexity from the verifier to the prover, but not from the prover to the verifier (hence only saving us a constant factor in communication complexity).

Second, communication complexity, while important, is always a secondary constraint in the area of interactive coding, as compared to achieving *constant* error-resiliency α. This is clearly recognized in the literature, as, for example, Ghaffari, Haeupler, and Sudan [14] show how to achieve optimal $\alpha = 2/7$ for traditional ("completeness-only") interactive coding, at the expense of quadratic blow-up in communication complexity.

Third (and related), it is certainly not obvious how to achieve any constant error-resiliency α, *irrespective of the communication overhead allowed.* Indeed, the trivial technique of applying error-correcting codes to every message will only achieve $\alpha = O(1/r)$, where r is the number of communication rounds. In contrast, our work achieves (optimal) $\alpha = \Omega(1)$, even while maintaining soundness.

Finally, the seemingly large (but polynomial) communication complexity blow-up is largely a matter of a rather arbitrary historical tradition defining the class **IP** as having a *constant* soundness error. Traditionally, this was always justified by the parallel repetition, even though such repetition only reduces the soundness error at the expense of the communication complexity! To see this more clearly, imagine an alternative definition of **IP**, where the soundness error is $2^{-\Omega(r)}$ (where r is the round complexity). While quantitatively different, it clearly does not change the resulting class **IP** (due to parallel repetition). Yet, with this (qualitatively equivalent!) definition we only need to run parallel repetition a *constant* number of times to gain the extra factor $2^{-\Omega(r)}$ needed to make our protocol backtracking-resilient. This means that our compiler would suddenly become "asymptotically optimal" even for communication complexity, even though nothing really changed from the conceptual point of view!

To summarize, unlike the goal of achieving constant error-rate α, the question of achieving better communication complexity blow-up seems to be less well motivated and largely dependent on rather arbitrary definitional choices. Still, once constant α is achieved by our work, it is an interesting open problem if $O(r)$ communication overhead is inherent using the specific (constant soundness) variant of **IP** that that we utilized following a historical tradition.

Error-Resilient Arguments. Another interesting direction is to add error-resilience to arguments, where the soundness condition only holds against a computationally sound prover. At first glance, this appears trivial, since our compiled protocol has honest prover/verifier which are efficient relative to the original prover/verifier. The subtlety comes from the fact our compiler must amplify the computational soundness of the original argument from 1/2 to $2^{-\Omega(r)}$. For proofs, such amplification is trivial via parallel repetition. In contrast, hardness amplification for arguments must involve an *explicit reduction.* And although many such reductions exist [21, 18, 8] for public-coin arguments, all of them come at the expense of a horrible degradation in the running time of the malicious prover. In particular, for polynomially bounded provers these reductions can "only" amplify soundness to become negligible in the security parameter, which is not enough to absorb a factor $2^{O(r)}$ we need, when the number of rounds r is polynomial in the security parameter. Moreover, Dodis et al. [10] gave strong evidence that hardness amplification "beyond negligible" is false in general, suggesting that a radically new approach is required for adding error-resilience to arguments.

Other Security Properties? Finally, given the negative results of [9, 12] for zero-knowledge/privacy in the presence of adversarial noise, coupled with our positive results for soundness, it is interesting to characterize which other security properties can withstand adversarial errors, and at what cost.

References

[1] Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. manuscript, 2013. `http://arxiv.org/abs/1312.4182`.

[2] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[3] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. In *Computational Complexity, Vol. 3, No. 4*, pages 319–354, 1993.

[4] Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *FOCS*, pages 160–166, 2012.

[5] Mark Braverman. Towards deterministic tree code constructions. In *ITCS*, pages 161–167, 2012.

[6] Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *FOCS*, 2014.

[7] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *STOC*, pages 159–166, 2011.

[8] Kai-Min Chung and Feng-Hao Liu. Parallel repetition theorems for interactive arguments. In *TCC*, pages 19–36, 2010.

[9] Kai-Min Chung, Rafael Pass, and Siddartha Telang. Knowledge-preserving interactive coding. In *FOCS*, 2013.

[10] Yevgeniy Dodis, Abhishek Jain, Tal Moran, and Daniel Wichs. Parallel repetition theorems for interactive arguments. In *TCC*, pages 467–493, 2012.

[11] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *FOCS*, pages 768–777, 2011.

[12] Ran Gelles, Amit Sahai, and Akshay Wadia. Private interactive communication across an adversarial channel. In *ITCS*, pages 135–144, 2014.

[13] Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding ii: Efficiency and list decoding. In *FOCS*, 2014.

[14] Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding: Adaptivity and other settings. In *STOC*, 2014.

[15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[16] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, pages 59–68, 1986.

[17] Bernhard Haeupler. Interactive channel capacity revisited. In *FOCS*, 2014.

[18] Johan Hastad, Rafael Pass, Krzysztof Pietrzak, and Douglas Wikstrom. An efficient parallel repetition theorem for arthur-merlin games. In *TCC*, pages 1–18, 2010.

[19] Cristopher Moore and Leonard J. Schulman. Tree codes and a conjecture on exponential sums. *CoRR*, abs/1308.6007, 2013.

[20] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[21] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. An efficient parallel repetition theorem for arthur-merlin games. In *STOC*, pages 420–429, 2007.

[22] Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *FOCS*, pages 724–733, 1992.

[23] Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC*, pages 747–756, 1993.

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

[25] Adi Shamir. Ip=pspace. In *FOCS*, pages 11–15, 1990.