Knowledge-Preserving Interactive Coding

Kai-Min Chung
Academia Sinica, Taiwan
kmchung@iis.sinica.edu.tw

Rafael Pass
Cornell University
rafael@cs.cornell.edu

Sidharth Telang
Cornell University
sidtelang@cs.cornell.edu

***Abstract*—How can we encode a communication protocol between two parties to become resilient to adversarial errors on the communication channel? If we encode each message in the communication protocol with a "good" error-correcting code (ECC), the error rate of the encoded protocol becomes poor (namely $O(1/m)$ where m is the number of communication rounds). Towards addressing this issue, Schulman (FOCS'92, STOC'93) introduced the notion of *interactive coding*.**

We argue that whereas the method of separately encoding each message with an ECC ensures that the encoded protocol carries the same amount of information as the original protocol, this may no longer be the case if using interactive coding. In particular, the encoded protocol may completely leak a player's private input, even if it would remain secret in the original protocol. Towards addressing this problem, we introduce the notion of *knowledge-preserving interactive coding*, where the interactive coding protocol is required to preserve the "knowledge" transmitted in the original protocol. Our main results are as follows.

- **The method of separately applying ECCs to each message has essentially optimal error rate: No knowledge-preserving interactive coding scheme can have an error rate of $1/m$, where m is the number of rounds in the original protocol.**
- **If restricting to computationally-bounded (polynomial-time) adversaries, then assuming the existence of one-way functions (resp. subexponentially-hard one-way functions), for every $\epsilon > 0$, there exists a knowledge-preserving interactive coding schemes with constant error rate and information rate $n^{-\epsilon}$ (resp. $1/\mathrm{polylog}(n)$) where n is the security parameter; additionally to achieve an error of even $1/m$ requires the existence of one-way functions.**
- **Finally, even if we restrict to computationally-bounded adversaries, knowledge-preserving interactive coding schemes with constant error rate can have an information rate of at most $o(1/\log n)$. This results applies even to *non-constructive* interactive coding schemes.**

Chung is supported by NSF Award CNS-1217821, NSF Award CCF-1214844 and Pass' Sloan Fellowship; work was done when being at Cornell.

Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2- 0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

I. Introduction

The study of how to communicate over a noisy channel dates back to the seminal works of Shannon [Sha48] and Hamming [HAM50] from the 1940s, initiating the study of error-correcting codes. Roughly speaking, an error-correcting code encodes a k-bit message m into a ck bit message with the property that even if a fraction $\eta < 1$ of the bits of the encoded messages are adversarially changed, the original message m can be decoded; $R = 1/c$ is referred to as the *information rate* of the code, and η as the *error rate*. Efficiently encodable and decodable error correcting codes with constant information rate and error rate are known [Jus72]; in fact, such codes can even be made linear-time encodable and decodable [Spi96].

In this work, we are interested in the question of how to encode interactive communication: Given an interactive protocol $\pi = (A, B)$ between two parties, how can we encode this protocol to become resilient to adversarial errors? A naive approach would be to simply apply a "good" (i.e., constant information and error rate) error correcting code to each message of the protocol. This results in a poor error rate: if the protocol has m rounds and each round requires sending a k-bit message, then it suffices to corrupt $O(k)$ out of the $O(km)$ communicated bits (that is, a fraction $O(1/m)$) to ensure an incorrect decoding. To address this problem, Schulman [Sch92], [Sch93], [Sch96] introduced the notion of *interactive coding*. Roughly speaking, an interactive coding scheme is an algorithm $Q = (Q_1, Q_2)$ such that for any interactive protocol $\pi = (A, B)$, (Q_1^A, Q_2^B) emulates the interaction of (A, B) in the sense that (with overwhelming probability) the execution of the actual protocol (A, B) and the "encoded protocol" (Q_1^A, Q_2^B) yield the same outputs. Additionally, the protocol (Q_1^A, Q_2^B) is error-resilient: the execution of (Q_1^A, Q_2^B) yields the same outputs even if a η fraction of the communication is adversarially corrupted, where η is an error rate. Schulman [Sch96] presented an interactive coding scheme with constant information and error rate. Schulman's construction achieved an error rate of $1/240$, which was later improved by Braverman and Rao [BR11] and Braverman [Bra12] to (close to) $1/8$. The interactive coding scheme

Q in their works, however, requires exponential or subexponential time. Gelles, Moitra and Sahai [GMS11] showed to get a polynomial-time interactive coding (with constant information and error rate) for the case of uniformly distributed (as opposed to adversarial) errors. More recently, the elegant work of Brakerski and Kalai [BK12] showed how to get a polynomial-time interactive coding handling also adversarial errors, with a constant information rate and an error rate of (close to) $1/32$, and even more recently Brakerski and Naor [BN13] show how to get quasi-linear time interactive coding with constant information and error rate.

Interactive Coding, revisited When we encode messages using error correcting codes we ensure that the encoded messages carry exactly the same information as the original messages; in other words, they carry all the information in the original messages (or else we cannot decode), and additionally they do not carry any other information (say, about future messages). Consider, for instance, transmitting an "interactive exam" (e.g., an oral exam) in an error resilient way. The exam has the property that question 2 in the exam reveals the answer to question 1. Ideally, we would like to guarantee that the error resilient version of the exam does not allow the student (taking the exam) to see question 2 before it needs to provide the answer to question 1 (or else it can trivially answer question 1). Clearly this property would hold if we use the "naive approach" of separately encoding every message using an error correcting code, but as we shall see shortly, this property may no longer hold if we use interactive coding. Intuitively, the problem is that interactive coding (and in particular, the above-mentioned solutions), while guaranteeing that the encoded protocol carries at least the same amount of information as the original protocol, does not necessarily guarantee that the encoded protocol does not reveal more information than the original protocol.

As another example, consider two mutually distrustful players that wish to runs some secure cryptographic protocol (A, B) over a noisy channel. Can these players instead run an interactive coding (Q_1^A, Q_2^B) of (A, B)? In other words, does the interactive coding preserve the security of the original underlying protocol? It is easy to see that the "naive approach" of separately encoding every message using an error correcting code preserves security of the underlying protocol. However, if we use interactive coding, this may no longer be the case. The problem is that the notion of interactive coding only requires that the encoded protocol (Q_1^A, Q_2^B) emulates (A, B) as long as both of the communicating parties are *honestly executing* the protocol. In particular, if one of the players is adversarial, it could be the case that the player gains more information when participating in the encoded protocol than it would have in the original protocol; for instance, player 1 may, by deviating from the protocol instructions in the encoded protocol (Q_1^A, Q_2^B), learn something about player 2's private input that is guaranteed to remain secret in the original protocol (A, B) (no matter what player 1 does).

The reason interactive coding schemes do not necessarily provide the desired guarantees in the above scenarios is that such schemes typically "bundle together" multiple rounds of interactions of the original protocol, and when an error in the communication is detected, the whole bundle is "replayed". (Looking forward, as we show in Theorem 2, any interactive coding with a "good" error rate in fact needs to replay messages in this way.) This may allow an attacker to "fake" an error in the communication in order to get the bundle replayed, but this time change its messages, and as a consequence may learn two (or more) partial transcripts where the attackers messages are different: in essence, the encoded protocol gives the attacker the opportunity to "rewind" the honest player in original protocol. In the above "interactive exam" example such rewindings mean that the student may get knowledge of the second question before having to provide the answer to the first one; for the cryptographic protocol example, it is well-known that most (but not all, see [CGGM00]) cryptographic protocols are not secure under such rewindings: Consider, for instance, any of the classic zero-knowledge protocols (e.g., [GMR89], [Blu86]); if the verifier can rewind the prover just once, it can completely recover the NP-witness used by the prover, although the protocols are zero-knowledge without such rewindings.

Knowledge-preserving interactive coding Towards addressing this problem, we here put forward, and study, the notion of *knowledge-preserving interactive coding*: Roughly speaking, we require not only that (Q_1^A, Q_2^B) conveys at least as much "knowledge" as (A, B), but also that it does not convey more, even if one of the players adversarially deviates from the protocol instructions; that is, (Q_1^A, Q_2^B) preserves the knowledge transmitted in (A, B). In other words, we require not only that (Q_1^A, Q_2^B) emulates (A, B) when the players are honest (only caring about their correct output and not trying to extract any other knowledge), but also that it is a good emulation when one of the players adversarially deviates from the protocol instructions (e.g., trying to obtain more knowledge about the other player's input and potentially use it in the interaction). We formalize this notion through the classic *zero-knowledge* "simulation-paradigm" from cryptography [GMR89], [GMW91]: We require that

for every adversarial strategy $\tilde{A}^*$ for player 1 (resp. $\tilde{B}^*$ for player 2) participating in the encoded protocol $(\tilde{A}, \tilde{B}) = (Q_1^A, Q_2^B)$, there exists a "simulator" A^* (resp. B^*) such that the output of both players in the execution of $(\tilde{A}^*, \tilde{B})$ (resp. $(\tilde{A}, \tilde{B}^*)$) are indistinguishable from the outputs of players in the execution of (A^*, B) (resp. (A, B^*)). In other words, an adversary participating in the encoded protocol does not gain any more "knowledge" than it would have in the original protocol, and cannot affect the honest parties output more than it could have in the original protocol.

As we shall see, achieving knowledge-preserving interactive coding is significantly harder than "plain" interactive coding, and studying *resilience against only computationally bounded adversaries*, as was done by Lipton [Lip94] and Micali, Peikert, Sudan and Wilson [MPSW10] in the context of error correcting codes, is actually *essential* for achieving good error rates in the context of knowledge-preserving interactive coding.

A. Our Results

We are interested in knowledge-preserving interactive coding schemes $Q = (Q_1, Q_2)$ where Q_1 and Q_2 are efficient; we formalize this by requiring that Q_1, Q_2 receive as input the communication complexity ℓ and number of rounds m of the protocol (A, B), and a security parameter n, and require that Q_1, Q_2 run in time polynomial in ℓ, m and n; in the sequel, when referring to a knowledge-preserving interactive coding scheme, we only refer to such efficiently computable schemes.

The information-theoretic regime We start by stating the folklore result that the "naive approach" of separately encoding each message in the protocol with a good error correcting code is a knowledge-preserving interactive coding:

Theorem 1. *[Informally stated] There exists a knowledge-preserving interactive coding scheme Q with polynomial information rate and error rate $O(1/m)$ where m is the number of communication rounds in the original protocol.*

Our first result is a strong negative result for knowledge-preserving interactive coding, showing that the naive approach is essentially optimal in terms of the error-rate (if requiring resilience against computationally unbounded adversaries).[1]

Theorem 2. *[Informally stated] For every knowledge-preserving interactive coding scheme $Q = (Q_1, Q_2)$, every polynomial $m(\cdot)$, there exists an $m(n)$-round protocol (A, B) such that (Q_1^A, Q_2^B) has an error rate of at most $1/m(n)$, where n is the security parameter. (In particular, no knowledge preserving coding scheme can have error rate $1/\mathrm{poly}(n)$ where n is the security parameter).*

[1]We mention the naive approach also has a poor (polynomial) *information* rate. We leave open the question if constant information rate can be acheived with error rate $O(1/m)$.

The computational regime We next turn to consider *computational knowledge-preserving interactive coding*, where we only require resilience against computationally bounded adversaries: we only require the error-resilience property to hold against computationally-bounded channel adversaries, and the knowledge-preserving property to hold against computationally-bounded adversaries. We first present a positive result, showing that constant-error rate is possible (albeit at a sub-constant information rate):

Theorem 3. *[Informally stated] Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists a knowledge-preserving interactive coding scheme with error rate $(1/12) - \epsilon$ and information rate $O(1/n^\epsilon)$ where n is the security parameter. If additionally subexponentially-hard one-way functions exists, the information rate can be improved to $O(1/\mathrm{polylog} n)$.*

As our next result demonstrates, one-way functions are *necessary* to achieve a "non-trivial" error rate.

Theorem 4. *[Informally stated] Assume the existence of a computational knowledge-preserving interactive coding scheme with error rate $1/m$, where m is the number of communication rounds in the original protocol. Then one-way functions exist.*

We finally show that every computational knowledge-preserving interactive coding scheme with constant error rate must have an information rate of $o(1/\log n)$.

Theorem 5. *[Informally stated] Assume the existence of a computational knowledge-preserving interactive coding scheme with information rate R and error rate η. Then $R\eta \in o(1/\log(n))$, where n is the security parameter.*

Non-constructive knowledge-preserving interactive coding All the above-mentioned results rely on the standard notion of interactive coding where the algorithm $Q = (Q_1, Q_2)$ only uses the original protocol $\pi = (A, B)$ as a black-box (i.e., the encoded protocol is (Q_1^A, Q_2^B)). One may also consider a more relaxed notion of coding, where the encoded protocol uses the description of the protocol π in a *non-black-box* way, or is even *non-constructive*. We note that the proof of Theorem 5 is actually stronger than stated; we actually show that *every* protocol (not just those protocols obtained by accessing the original protocol π as a black-box) that preserves the knowledge transmitted in the

original protocol and has an error rate of $O(1)$, must have a communication complexity of at least $\omega(\log n)$.

Theorem 6. *[Stronger version of Theorem 5, informally stated] For every function $\eta(n) \in \Omega(1/\log n)$, there exists a protocol π with communication complexity $O(1/\eta(n))$ such that for every protocol π' that is a knowledge-preserving variant of π (even just w.r.t. computationally-bounded adversaries) and is computationally η-error resilient, the communication complexity of π' is at least $\omega(\log n)$.*

It is worthwhile to compare Theorem 6 with Theorem 2. As mentioned above, Theorem 6 is stronger that Theorem 2 in that it rules out also non-constructive interactive coding schemes. On the other hand, it is weaker is several other aspects: First, in Theorem 2 we "blatantly" violate knowledge preservance: we exhibit some explicit information that can only be learnt with negligible probability in π, but can be learnt with inverse polynomial probability in the encoded protocol. In contrast, in the proof of Theorem 6 we rely on the knowledge-preservance property in a stronger way (in particular, as mentioned above, we rely the knowledge-preservance property to show that the encoded protocol implicitly executed π, whereas in Theorem 2 this could be showed unconditionally). Secondly, the error rate achieved in Theorem 6 is weaker than the one rate achieved in Theorem 2. This, to some extent, is necessary, since Theorem 6 also rules out computational knowledge-preserving interactive coding, and as showed in our positive result (Theorem 3), an error rate of $O(1)$ can be achieved in this setting.

B. Independent Work

In this work we study whether interactive codings schemes can be made knowledge preserving (and thus preserving security of the protocols on which they operate). A recent independent work by Gelles, Sahai and Wadia [GSW13] focuses on an orthogonal, but related, exciting new question and studies whether information-theoretically secure computation protocols can be error resilient. They provide a negative result by demonstrating the existence of a class of functionalities that can be securely computated with information-theoretic security, but no error-resilient protocol (handling a constant error rate) can compute these functionalities with information theoretic-security.

C. Overview of the Paper

In Section II we provide some notation and preliminaries. In Section III we formally define the notion of knowledge-preserving interactive coding. Section IV contains our results for the information-theoretic setting, and Section V contains our result for the computational setting; finally, in Section VI we present our impossibility results for non-constructive interactive coding. Due to lack of space, in this extended abstract we only provide very rough high-level outlines of the proofs of all our results. The complete proofs can all be found in the full version [CPT13].

II. Notation and Preliminaries

A. Notation

Basic Notation Let $\mathbb{N}$ denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \ldots, n\}$. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, the notation "$M_r(x)$" denotes the output of the M on input x when M's random tape is fixed to r, while $M(x)$ represents the distribution of outputs of $M_r(x)$ when r is chosen uniformly. We say that a function $\epsilon : \mathbb{N} \to [0,1]$ is *negligible* if for every constant $c \in \mathbb{N}$, $\epsilon(n) < n^{-c}$ for sufficiently large n. We say that a function $\mu : \mathbb{N} \to [0,1]$ is *overwhelming* if there exists a negligible function ϵ such that for all $n \in \mathbb{N}$, $\mu(n) \geq 1 - \epsilon(n)$.

Probabilistic Notation We use probabilistic notation from [GMR89]: By $x \leftarrow \mathcal{S}$, we denote that an element x is sampled from a distribution $\mathcal{S}$. If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F . To denote the ordered sequence in which the experiments happen we use comma, e.g. $(x \leftarrow \mathcal{S}, (y,z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot,\cdot)$ denotes a predicate, then $\Pr[x \leftarrow \mathcal{S}, (y,z) \leftarrow A(x) : p(y,z)]$ is the probability that the predicate $p(y,z)$ is true in the ordered sequence of experiments $(x \leftarrow \mathcal{S}, (y,z) \leftarrow A(x))$. The notation $\{(x \leftarrow \mathcal{S}, (y,z) \leftarrow A(x) : (y,z))\}$ denotes the resulting probability distribution $\{(y,z)\}$ generated by the ordered sequence of experiments $(x \leftarrow \mathcal{S}, (y,z) \leftarrow A(x))$.

Notation for Interactive protocols An interactive protocol is a tuple $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ where A and B are interactive probabilistic Turing machines and $\mathcal{X}_A$ and $\mathcal{X}_B$ specify the set of inputs to A and B (parametrized by a security parameter n). A and B, on input $x \in \mathcal{X}_A(1^n)$ and $y \in \mathcal{X}_B(1^n)$ interact with each other and generate some output at the end of the interaction. We denote this interaction by $A(x) \leftrightarrow B(y)$ (formally, it is a random variable over joint views of A and B, including the randomness of both players, their inputs, and all the messages received). Given an interaction e, we denote by $\mathsf{out}_i[e]$ the output of player $i \in \{1,2\}$, by $\mathsf{out}[e]$ the output of both parties, and by $\mathsf{trans}[e]$ the transcript of the interaction.

B. Cryptographic Notions

We assume the reader is familiar with statistical and computational indistinguishability, one-way functions and signature schemes; we refer the reader to [Gol06] (or the full version of this paper) for details.

C. Hash Functions

We recall the standard definition of t-wise independent hash functions.

Definition 7 (t-wise Independent Hash Functions)**.** A family of hash functions $H = \{h : S_1 \to S_2\}$ is *t-wise independent* if the following two conditions hold:

1) $\forall x \in S_1$, the random variable $h(x)$ is uniformly distributed over S_2, where $h \leftarrow H$.
2) $\forall x_1 \neq \cdots \neq x_t \in S_1$, the random variables $h(x_1), \ldots, h(x_t)$ are independent, where $h \leftarrow H$.

D. Error-correcting Codes

We recall the definition of error correcting codes.

Definition 8 (Coding function)**.** An *(n, ℓ)-coding function* $C = (E, D)$ is an encoding function $E : \{0,1\}^n \to \{0,1\}^\ell$ and a decoding function $D : \{0,1\}^\ell \to \{0,1\}^n$ for some positive integers $\ell \geq n$. The *information rate* of the scheme, denoted R is defined as n/ℓ. The scheme has *error rate (or decoding distance)* η if, for all $m \in \{0,1\}^n$ and all $r \in \{0,1\}^\ell$ such that the *codeword* $E(m)$ and r differ in at most $\eta\ell$ bits, $D(r) = m$.

Definition 9 (ECC)**.** An family of coding functions $\{C_n = (E_n, D_n)\}_{n\in\mathbb{N}}$ is an *efficient error correcting code (ECC)* $C = (E, D)$ with *error rate* $\eta : \mathbb{N} \to (0, 1)$ and *information rate* $R : \mathbb{N} \to (0, 1)$ if for every $n \in \mathbb{N}$, (E_n, D_n) is a $(n, n/R(n))$ coding function with error rate $\eta(n)$, and $\{E_n\}$ and $\{D_n\}$ can be computed by uniform polynomial time algorithms.

As shown by Justesen in [Jus72], ECCs with constant error and information rate exist.

Theorem 10 ([Jus72])**.** *There exists an ECC with error rate $O(1)$ and information rate $O(1)$.*

Justesen codes, however, do not give a tight error rate. The concatenation of the Reed-Solomon code [RS60] and the Hadamard code, however, yield an ECC with error rate close to $1/4$ (which is optimal), but require a polynomial information rate.

Theorem 11 ([GS00])**.** *For every $\epsilon > 0$, there exists an ECC with error rate $\frac{1}{4} - \epsilon$ and information rate $R(n) = O(1/n)$.*

When unique decoding is impossible, it may be still possible to decode to a short list of candidate messages; the notion of list-decoding captures this.

Definition 12. A (n, ℓ)-coding function is *(ϵ, L)-list decodable* if for any $r \in \{0,1\}^\ell$, there exists a list of at most $l \leq L$ distinct $m_1, m_2, \ldots m_l$ such that $E(m_i)$ and r differ in at most $\epsilon\ell$ bits, for all $i \in [l]$. A family of coding functions $\{(E_n, D_n)\}_{n\in\mathbb{N}}$ with *information rate* $R : \mathbb{N} \to (0, 1)$ is *efficiently ϵ-list decodable* if for every $n \in \mathbb{N}$, (E_n, D_n) is a $(n, n/R(n))$ coding function that is $(\epsilon(n), L_n)$-list decodable for some $L_n \in \mathbb{N}$, and there exists a polynomial time algorithm LD that finds this list.

As shown by Guruswami and Sudan [GS00], the concatenation of the Reed-Solomon code and Hadamard code is efficiently list decodable up to an error rate close to $1/2$.

Theorem 13. *For every $\epsilon > 0$, there exists an ECC with information rate $R(n) = O(1/n)$ that is is efficiently $\frac{1}{2} - \epsilon$-list decodable.*

III. KP Interactive Coding

In this section we provide a formal definition of knowledge-preserving (KP) interactive coding.

A. Error Resilience for Interactive Protocols

Let us start by defining *error rate* for interactive protocols; in contrast to earlier works on interactive coding, we here consider error-resilience against both unbounded adversarial channels (as is typically done [Sch96]) and also error-resilience against computationally bounded channels (as was done in the context of error correcting codes in [Lip94], [MPSW10]).

Towards providing these definitions, we need some additional notation. The *communication complexity* of a protocol π on security parameter n, denoted by $\mathrm{CC}_n(\pi)$, is the worst-case total number of bits transmitted in the interaction $A(x) \leftrightarrow B(y)$, over all possible input $x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$ and randomness of both players. The round complexity $m(n)$ of a protocol π on security parameter n is the worst-case number of communication rounds (one round corresponds to two messages) in the interaction $A(x) \leftrightarrow B(y)$, over all possible input $x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$ and randomness of both players.

We consider interactive protocols running over noisy/adversarial channels, which may flip some of the bits transmitted by both players. We model channels as interactive Turing machines that relay messages between the two players.

Definition 14 (Channel)**.** A channel is an interactive Turing machine C that on input a security parameter 1^n interacts with two interactive machines by relaying messages for the machines as follows: upon receiving a message m from one machine, C sends a message m' to the other machine of length $|m'| = |m|$.

We denote by $A(x) \leftrightarrow_{C(1^n)} B(y)$ the interaction between $A(x)$ and $B(y)$ over the channel C given the security parameter n. (Note that the interaction $A(x) \leftrightarrow B(y)$ is identical to the interaction $A(x) \leftrightarrow_{C_0(1^n)} B(y)$, where C_0 is the "honest" channel that simply relays messages between A and B without flipping any bits.)

Definition 15 (Communication Complexity over Noisy Channels)**.** Let $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ be a protocol. The *communication complexity of π over noisy channels* on security parameter n, denoted by $\mathrm{CC}^*_n(\pi)$, is the worst-case number of bits transmitted in the interaction $A(x) \leftrightarrow_{C(1^n)} B(y)$, over all possible inputs $x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$, randomness of both players, and channels C.

We are now ready to define *error resilience*.

Definition 16. A protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ with round-complexity $m(\cdot)$ is *(computationally) $\eta(\cdot,\cdot)$-error resilient* if there exists a negligible function μ such that for every security parameter n, inputs $x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$, and any (non-uniform probabilistic polynomial-time in the security parameter n)[2] channel C that flips at most $\eta(n, m(n)) \cdot CC^*_n(\pi)$ bits, the following holds:

$$\Pr\left[\mathsf{out}[A(x) \leftrightarrow B(y)] = \mathsf{out}[A(x) \leftrightarrow_{C(1^n)} B(y)]\right] \geq 1 - \mu(n).$$

B. Knowledge Preservance

Let us move on to defining what it means for a protocol $\tilde{\pi} = (\tilde{A}, \tilde{B}, \mathcal{X}_A, \mathcal{X}_B)$ to be convey "as much knowledge" as a protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$. We formalize this using the classic "simulation-paradigm" from cryptography [GMR89], [GMW91]. We require that for every adversarial strategy $\tilde{A}^*$ for player 1 (resp. $\tilde{B}^*$ for player 2) participating in $\tilde{\pi}$, there exists a simulator A^* such that the output of both players in the execution of $(\tilde{A}^*, \tilde{B})$ (resp. $(\tilde{A}, \tilde{B}^*)$) are indistinguishable from the outputs of players in the execution of (A^*, B) (resp. (A, B^*)). That is, any "harm" $\tilde{A}^*$ can do in $\tilde{\pi}$, the simulator A^* could have also done in π.

Definition 17. A protocol $\tilde{\pi} = (\tilde{A}, \tilde{B}, \mathcal{X}_A, \mathcal{X}_B)$ is a *(computationally) knowledge-preserving variant of* $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ if the following two properties hold:

- *Completeness:* There exists a negligible function μ such that the following ensembles are statistically close as a function of n.
  - $\{\mathsf{out}[\tilde{A}(x) \leftrightarrow \tilde{B}(y)]\}_{n,x,y}$
  - $\{\mathsf{out}[A(x) \leftrightarrow B(y)]\}_{n,x,y}$

  where the ensembles are indexed over $n \in \mathbb{N}, x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$.
- *(Computational) Knowledge Preservance:* For every (probabilistic polynomial-time) adversary strategy $\tilde{A}^*$ for player 1, there exists a (probabilistic polynomial-time) strategy A^* such that the following ensembles are statistically close (resp. computationally indistinguishable) as a function of n.
  - $\{\mathsf{out}[\tilde{A}^*(x, z) \leftrightarrow \tilde{B}(y)]\}_{n,x,y,z}$
  - $\{\mathsf{out}[A^*(x, z) \leftrightarrow B(y)]\}_{n,x,y,z}$

  where the ensembles are indexed over $n \in \mathbb{N}, x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n), z \in \{0,1\}^*$. We make the analogous requirement for every (probabilistic polynomial-time) adversary strategy $\tilde{B}^*$ for player 2.

A Remark on Auxiliary Input Just as in the classic definitions of zero-knowledge [GMR89], [GO94] and secure computation [GMW91], the additional input z to $\tilde{A}^*$ (and A^*) models any auxiliary information available to the attacker. All our results hold regardless of whether we allow the attacker to receive such auxiliary information.

A Remark on Preserving Cryptographic Protocol Security In this comment we assume the reader is familiar with classic definitions of protocol security [GMW91]; see [Gol04] for details. It easily follows from the definition of knowledge preservance that if a protocol π is a "secure implementation" of some functionality $\mathcal{F}$ (in the sense of [Gol04]), then any knowledge-preserving variant π' of π will also be a secure implementation of $\mathcal{F}$. Indeed, if π' is a knowledge-preserving variant of π, then π' is "as secure as" π.

C. Knowledge-Preserving Interactive Coding

We are now ready to define *knowledge-preserving interactive coding*. An *interactive coding scheme* is a pair of oracle-aided interactive probabilistic Turing machines $Q = (Q_1, Q_2)$. For every interactive protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$, Q induces an *encoded interactive protocol* $Q^\pi = (Q_1^A, Q_2^B, \mathcal{X}_A, \mathcal{X}_B)$, defined as follows. In the interaction of Q^π, Q_1 and Q_2 do not receive the input directly. Instead, Q_1 and Q_2 receive as input the security parameter 1^n, the round complexity 1^m and the communication complexity $1^{CC_n(\pi)}$ of π, and are given oracle access to $A_{r_A}(x)$ and $B_{r_B}(y)$ respectively, where $x \in \mathcal{X}_A(1^n), y \in \mathcal{X}_B(1^n)$ are the inputs and $r_A, r_B \in \{0,1\}^\infty$ are uniformly sampled. More precisely, Q_1 (resp., Q_2) gets oracle access to the next-message functions of $A_{r_A}(x)$ (resp., $B_{r_B}(y)$), which on input a partial transcript T returns the next message (or the final output, in case T is a complete transcript). The interaction is denoted by $Q_1^{A(x)} \leftrightarrow Q_2^{B(y)}$ where the inputs 1^n, 1^m, and $1^{CC_n(\pi)}$ are omitted for notational

[2] We could also have defined the channel as a *uniform* polynomial-time algorithm. All our result hold for both choices.

simplicity. When we are explicit about the randomness used by Q_1 and Q_2, we write $Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_2)$.

Definition 18 (Knowledge-Preserving Interactive Coding Schemes)**.** Let $\eta(\cdot,\cdot), r(\cdot,\cdot) \in (0,1)$ be functions. A pair of oracle-aided interactive probabilistic Turing machines $Q = (Q_1, Q_2)$ is a *(computational) knowledge-preserving interactive coding scheme with error rate* $\eta(\cdot,\cdot)$ *and information rate* $R(\cdot,\cdot)$ if for every interactive protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$, the corresponding encoded protocol Q^π satisfies the following properties.

- *Efficiency*: Q_1 and Q_2 run in polynomial time in $n, m(n)$ and $\mathrm{CC}_n(\pi)$.
- *Information Rate:* $\mathrm{CC}^*_n(Q^\pi) \leq \mathrm{CC}_n(\pi)/R(n, m(n))$; that is the worst-case "blow-up" of the encoded protocol is bounded by $1/R(n, m(n))$.
- *Error Resilience:* Q^π is (computationally) η-error resilient.
- *Knowledge Preservance:* Q^π is a (computationally) knowledge-preserving variant of π.

Q is a *(computational) knowledge-preserving interactive coding scheme with information rate* $R(\cdot)$ *and error rate* $\eta(\cdot)$ if Q is (computational) knowledge-preserving interactive coding scheme with information rate $R'(n,m) = R(n)$ and error rate $\eta'(n,m) = \eta(n)$.

IV. The Information-Theoretic Regime

As we shall see, achieving knowledge-preserving interactive coding is significantly harder than "plain" interactive coding, and studying *resilience against only computationally bounded adversaries* (as was done in [Lip94], [MPSW10] in the context of error correcting codes) actually is *essential* for achieving good error rates in the context of knowledge-preserving interactive coding.

To put our result in context, let us start by showing that the "naive approach" of separately encoding each message in the protocol with a good error correcting code is a knowledge-preserving interactive coding:

Theorem 19. *There exists a knowledge-preserving interactive coding scheme Q with information rate $R(n,m) = O(m)$ and error rate $\eta(n,m) = O(1/m)$.*

Proof: We simply pad each message in the protocol π to become of equal length (this increases the communication complexity by at most a factor m) and next encode each message using a constant-rate error correcting code; let $\tilde{\pi}$ denote the encoded protocol. Clearly, $\tilde{\pi}$ is error resilient as long as we corrupt less than one message; thus we have an error rate of $O(1/m)$. It easily follows that $\tilde{\pi}$ is a security preserving variant of π; the simulator A^* for an attacker $\tilde{A}^*$ for player 1 emulates an execution of $\tilde{\pi}$ for $\tilde{A}^*$ by simply encoding all messages in π (using the error correcting code) and decoding all messages received by $\tilde{A}^*$ before sending them to player 2. The simulator for player 2 is defined analogously. ■

Let us now turn to our main impossibility result for the information-theoretic setting. We show that naive approach is essentially optimal: namely, any knowledge preserving interactive coding scheme must have an error rate of at most $1/m$.

Theorem 20. *Let Q be a knowledge-preserving interactive coding scheme with information rate $R(\cdot,\cdot)$ and error rate $\eta(\cdot,\cdot)$. Then for every polynomial $m(\cdot)$, we have that for sufficiently large n, $\eta(n, m(n)) < 1/m(n)$. (In particular, there does not exists a knowledge-preserving interactive coding scheme with error rate $\eta(n) = 1/\mathrm{poly}(n)$.)*

Proof: Due to lack of space we provide just a high-level overview of the proof of the theorem. (The full proof can be found in the full version of the paper). The key idea is to come up with a protocol π having the property that the only way to make the protocol error resilient makes it possible for an attacker to "rewind" the honest players (just as what is done in known interactive protocols, as described above). Consider some interactive coding protocol $Q = (Q_1, Q_2)$ and let $M(n, m, \ell)$ be a polynomial upper bound on the number queries made by Q_1, Q_2 to its oracles (where n is the security parameter, m is the number of round in the protocol π to be encoded, and ℓ is the communication complexity of π). Consider the $m(n) = \mathrm{poly}(n)$-round "ping-pong" protocol π where each player $d \in \{1,2\}$ gets an $M(n, m, 2mn) + 1$-wise independent hash function $H_d : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n$ as input and proceeds as follows: player 1 computes and sends $a_1 = H_1(\emptyset)$ to player 2; player 2 computes and sends $b_1 = H_2(a_1)$ to player 1; player 1 computes and send $a_2 = H_1(a_1, b_1)$ to player 2, etc, for m rounds, and finally both player output the transcript of the interaction. That is, at each round, each player d, computes its next message by applying its hash function H_d to the current transcript. Note that in this protocol, by the unpredictability of the output of the hash functions, player 1, even if maliciously deviating from the protocol, will with overwhelming probability be able to obtain at most m distinct pairs $(q, H_2(q))$.

In contrast, as we show, unless the encoded protocol has an error rate less than $1/m$, a malicious player 1 in the encoded protocol can with inverse polynomial probability get $m+1$ such pairs (and as such a malicious player 1 can learn something new in Q^π that it couldn't have learnt in π). The key lemma needed to establish this shows that the encoded protocol "implicitly executes" the original ping-pong protocol. More precisely,

the rounds of the encoded protocol can be divided into "chunks", where each chunk in the encoded protocol corresponds to a round in ping-pong protocol[3], and additionally by observing the oracle queries made by Q, we can read out a polynomial list of candidates for the current transcript of the ping-pong protocol; to establish this lemma we rely on the "elusiveness" property of the output of the hash functions (and the fact that Q queries the hash functions at most $M(n, m, 2mn)$ times).

Next, by an averaging argument, one of these chunks, say chunk i, must be shorter than a fraction $1/m$ of the total communication complexity of the encoded protocol. The idea now is for a malicious player 1 to honestly execute the encoded protocol using its actual input, except that during the i'th chunk, the player acts as if its input was a random hash function H_1' consistent with the transcript up until the end of chunk $i-1$; that is, we switch the input only in chunk i, but make sure we pick an input that is consistent with the transcript so far. (Note that this attack is not necessarily efficient since picking an input consistent with the current transcript may not be computationally feasible.) Now, intuitively, since the chunk was "small", by the error resilience property of the interactive coding scheme, with overwhelming probability, player 1 will finally output the same transcript (including m distinct pairs $(q, H_2(q))$) as if it had been running the protocol honestly. (Formalizing this requires showing that the attack performed by player 1 can be perfectly emulated by the channel).

Additionally, as we show, by observing the oracle queries made by Q_1 during the i'th chunk (which corresponds to the i'th round in the implicitly executed ping-pong protocol), player 1 may learn a *new* pair $(q', H_2(q'))$; intuitively, this follows since player 1 is using a new input in round i of the implicitly executed ping-pong protocol (but formally proving this claim is quite non-trivial). Thus, in essence, player 1, by using a different input in only chunk i manages to "rewind" player 2 in the implicit ping-pong protocol.

So, if player 1 could just identify the i'th chunk, it can learn $m+1$ distinct pairs $(q, H_2(q))$, which was not possible in π; but it can simply guess the starting round of the i'th chunk with inverse polynomial probability. Summarizing, in π, an attacker can learn $m+1$ distinct pairs $(q, H_2(q))$ only with negligible probability, whereas in Q^π this can be done with inverse polynomial probability (if Q^π has a "non-trivial" error rate); thus, we are "blatantly" violating knowledge-preservance of Q. ■

[3]These chunks may depend on the inputs of the players and the randomness of Q.

V. The Computational Regime

We here turn to studying knowledge-preserving interactive coding in the presence of only computationally-bounded adversaries (i.e., computational knowledge-preserving interactive coding).

A. Positive Results

We first present a positive result, showing that assuming the existence of one-way function, computational knowledge-preserving interactive coding with constant error rate (more specifically, close to $1/12$) and sub-polynomial (or even poly logarithmic, if assuming subexponentially-hard one-way functions) information rate is possible.

Theorem 21. *Assume the existence of one-way functions. For every $\epsilon > 0$, there exists a computational knowledge-preserving interactive coding scheme with perfect completeness, error rate $\eta = \frac{1}{12} - \epsilon$ and information rate $R(n) = O(1/n^\epsilon)$. If additionally sub-exponentially hard one-way functions exists, the information rate is $1/\mathrm{polylog}n$.*

Proof: Again we simply provide a very high-level proof overview and defer the details to the full version. Roughly speaking, the idea behind the scheme is the following. In a "preamble phase" , the players start by exchanging verification keys for a signature scheme; the verification keys first are padded with ρ 0's to become "long" enough (where ρ is a parameter to be set) and then encoded using a good ECC $(E_\mathsf{p}, D_\mathsf{p})$; let $\alpha(n)$ denote the length of the encoded verification key. Next, in the "main execution phase" we run the original protocol π, except that each message is signed and encoded using a good error correcting code $(E_\mathsf{m}, D_\mathsf{m})$ (which may be different from the code $(E_\mathsf{p}, D_\mathsf{p})$). More precisely, player 1 keeps track of the "current round" number in the protocol π, and encode its i^{th}-round message a_i as $c_i = E_\mathsf{m}(i, a_i, \sigma_i)$ where σ_i is a signature of (i, a_i). Upon receiving a message c while having the "current round" number (in the protocol π) being i, player 1 decodes the message $((i', b), \sigma) = D_\mathsf{m}(c)$ and interprets b as the i'th round message b_i in π, as long as 1) $i' = i-1$, and 2) σ is a valid signature on (i', b); if not, player 1 "signals" that the received message was corrupted by simply resending its message $c_{i-1} = E_\mathsf{m}(i-1, a_{i-1}, \sigma_i)$ from the previous round. Player 2's strategy is defined analogously except that player 2 accepts the received message if $i' = i$ (since player 2 is sending the second message in each round). Finally, we impose a bound c on the communication complexity of the protocol (or else the protocol may run forever, due to "resend" message); both players simply abort outputting nothing if the communication complexity would exceed c if they send their message.

Let $\beta(n)$ denote the length of each encoded message, and let $\gamma(n,m) = 2\alpha(n) + 2m\beta(n)$ be the length of the protocol if all messages get sent through on the first trial, and there is no cut-off.

We must set ρ such that the length $\alpha(n)$ of the encoded verification keys is within a constant factor of c (or else, either the preamble phase can be fully corrupted, or the main phase can be fully corrupted). On the other hand, c must be long enough to execute the encoded version of π (that is $c > \gamma(n,m)$, and additionally handle sufficiently many "resend" requests, before the error-quota of the adversary runs out. By appropriately setting ρ and c, this leads to an error rate of $\eta/4$ if η is the error rate of both $(E_{\mathsf{p}}, D_{\mathsf{p}})$ and $(E_{\mathsf{m}}, D_{\mathsf{m}})$; roughly speaking, we lose a factor of two because the adversary may choose to corrupt either the verification key of player 1 or that of player 2; we loose another (additively) factor of two due to the fact that the length of the encoded messages must be within a constant factor of c, and the fact that each time the attacker corrupts the message of a single player in the main phase protocol execution, we need to resend the whole round (i.e., 2 messages).

We can further improve the error rate by relying on an idea from [MPSW10]: since the messages in the main phase are signed and we only consider a computationally bounded channel, it in fact suffices to list-decode the error-correcting code $(E_{\mathsf{m}}, D_{\mathsf{m}})$ used in the main phase.[4] It follows using the same argument as in [MPSW10] that (with overwhelming probability) list-decoding can only yields at most a single valid message (since all the messages are signed), and thus using list-decoding here yields unique decoding. ■

B. Negative Results

We show that our positive result is optimal in two ways:

1) The existence of one-way functions is necessary.
2) If a constant error rate is desired, it is impossible to achieve an information rate of $\Omega(1/\log n)$.

The necessity of one-way functions We show that the existence of one-way functions is necessary to achieve computational knowledge-preserving interactive coding with error rate $1/poly(n)$.

Theorem 22. *For every polynomial $m(\cdot)$, the existence of a computational knowledge-preserving interactive coding scheme Q with error rate $\eta(n,m) \geq 1/m(n)$ implies the existence of one-way functions.*

Proof: At a high level, the theorem follows by observing that the attacker in the proof of Theorem 20 can be approximated efficiently if one-way functions do not exist; the same holds for the channel adversary. The full proof is found in the full version. ■

The necessity of a communication complexity blow-up We show that every computational knowledge-preserving interactive coding scheme with constant error rate must have an information rate of $o(1/\log n)$, showing that the inverse polylogarithmic rate achieved in Theorem 21 (assuming subexponentially hard one-way functions) is essentially optimal.

Theorem 23. *Assume the existence of a computational knowledge-preserving interactive coding scheme with information rate $R(n)$ and error rate $\eta(n)$. Then $R(n)\eta(n) \in o(1/\log(n))$.*

Proof: Let us first mention that a weaker version of the above theorem, demonstrating that the information rate needs to sub constant (as opposed to $o(1/\log n)$) can be obtained by carefully "scaling down" the proof of Theorem 2 by considering a constant-round ping-pong protocol where the length of each message is $O(\log n)$. To give the tight bound, we need to rely on an even more scaled down version where the length of the messages in the ping-pong protocol is just 1. In this regime, the previous proof no longer works: we can no longer ensure that the transcript from the ping-pong protocol can be decoded by observing all the oracle calls made by Q. (In the proof of Theorem 2 this was proven by relying on the elusiveness property of the image of the hash function, but since we now consider the range $\{0,1\}$, this no longer holds.) Rather, we here provide a different information-theoretic definition of chunks and rely on the fact that the protocol is knowledge preserving to show that chunks are well-defined. To simplify the proof of this, we actually rely on a simpler variant of the ping-pong protocol, which we refer to as the "bit-exchange" protocol π (the protocol is almost identical to a protocol used in [CP11] in a quite different context): in the bit-exchange protocol, in round j, each player simply sends the j'th bit of its input.

The idea, which turn out to be quite subtle to formalize, is that if a partial transcript contained information about, say, player 2's message in round j, before player 1's message in round j has been fully determined in the partial transcript, then intuitively, player 1 has the opportunity to (with non-negligible probability) change its message in round j as a function of player 2's message in the same round, which isn't possible in the original bit-exchange protocol. The formal proof, which

[4] [MPSW10] relies on this idea to show how to achieve an error-correcting code with error rate $1/2 - \epsilon$ if assuming a (noiseless) public-key infrastructure and a computationally-bounded channel. In our context, we do not have a public-key infrastructure, but our initial exchange of verification keys using a uniquely decodable error-correcting code can be viewed as a way to set-up the appropriate public-key infrastructure needed for their results.

is quite subtle, is deferred to the full version. ■

VI. Non-constructive Schemes

We note that the proof of Theorem 5 is actually stronger than stated; we actually show that *every* protocol (not just those protocols obtained by accessing the original protocol π as a black-box) that preserves the knowledge transmitted in the "bit-exchange" protocol and has an error rate of $O(1)$, must have a communication complexity of at least $\omega(\log n)$.

As such, the lower bound applies even to *non-constructive* interactive coding scheme. In particular, for every $\eta(n) > 1/\log n$, we have demonstrated the existence of protocol π with communication complexity $1/\eta(n)$ such that *every* computationally η-error resilient, computationally knowledge-preserving variant of π must have communication complexity at least $\omega(\log n)$. In particular, for the case $\eta(n) = O(1)$, we get that the information rate also for non-constructive interactive coding is at most $o(1/\log n)$. (Note that this result is interesting also in the information theoretic setting, since in contrast to Theorem 20, we here provide an impossibility result also for non-constructive interactive coding.)

Theorem 24. *For every function $\eta(\cdot)$ such that $\eta(n) \geq O(1/\log n)$, there exists a protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ with communication complexity $\mathrm{CC}_n(\pi) = O(1/\eta(n))$ such that for every protocol $\pi' = (Q_1, Q_2, \mathcal{X}_A, \mathcal{X}_B)$ that is a computationally knowledge-preserving variant of π and is computationally η-error resilient, the communication complexity of π' is at least $\omega(\log n)$.*

References

[BK12] Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *FOCS*, pages 160–166, 2012.

[Blu86] M. Blum. How to prove a theorem so no one else can claim it. *Proc. of the International Congress of Mathematicians*, pages 1444–1451, 1986.

[BN13] Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *SODA '13*, 2013. To appear.

[BR11] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *STOC*, pages 159–166, 2011.

[Bra12] Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC '00*, pages 235–244, 2000.

[CP11] Kai-Min Chung and Rafael Pass. The randomness complexity of parallel repetition. In *FOCS*, pages 658–667, 2011.

[CPT13] Kai-Min Chung, Rafael Pass, and Sidharth Telang. Interactive coding, revisited. Cryptology ePrint Archive, Report 2013/160, 2013.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMS11] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *FOCS*, pages 768–777, 2011.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity, or all languages in np have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *STOC*, pages 181–190, 2000.

[GSW13] Ran Gelles, Amit Sahai, and Akshay Wadia. Private interactive communication across an adversarial channel. Cryptology ePrint Archive, Report 2013/259, 2013.

[HAM50] R. W. HAMMING. Error detecting and error correcting codes. *BELL SYSTEM TECHNICAL JOURNAL*, 29(2):147–160, 1950.

[Jus72] J. Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Trans. Inf. Theor.*, 18(5):652–656, Sep 1972.

[Lip94] Richard J. Lipton. A new approach to information theory. In *STACS*, pages 699–708, 1994.

[MPSW10] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction for computationally bounded noise. *IEEE Transactions on Information Theory*, 56(11):5673–5680, 2010.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society of Industrial and Applied Mathematics*, 8:300–304, 1960.

[Sch92] Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *FOCS*, pages 724–733, 1992.

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC*, pages 747–756, 1993.

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

[Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.