Knowledge-Preserving Interactive Coding

Kai-Min Chung
Cornell University
chung@cs.cornell.edu

Rafael Pass *
Cornell University
rafael@cs.cornell.edu

Sidharth Telang
Cornell University
sidtelang@cs.cornell.edu

August 26, 2013

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. $1/\text{polylog}(n)$) 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.

*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.

1 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 attack 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.

1.1 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).*

Let us provide a high-level overview of the proof of the theorem. 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

[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 $O(1/m)$.

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[2], 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.

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 $O(1/\text{polylog}n)$.*

The idea behind this scheme is simple: The players start by exchanging verification keys for a signature scheme; the verification keys are appropriately padded to become "long" and then

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

encoded using a good error-correcting code. Next, we run the original protocol, except that all messages in the protocol are signed and additionally encoded using a good error-correcting code. Whenever a player receives a message that does not have a valid signature, it requests to hear the message again. It is easy to show that this coding scheme is knowledge preserving (even against unbounded attackers); additionally, if the verification keys exchanged in the first round are long enough, then the scheme has error rate close to $\eta/4$, where η is the error rate of the error-correcting code. Using state of the art error-correcting codes this would yield an error rate of $1/16 - \epsilon$ (where $\epsilon > 0$ is an arbitrarily small constant). We can further improve the error rate by relying on an idea from [MPSW10]: since messages are signed and we only consider a computationally bounded channel, it in fact suffices to "list-decode" the error-correcting code used to encode the messages in the protocol (while still using unique decoding for error-correcting code used to encode the verification keys).[3] This allows us to improve the error rate to $1/12 - \epsilon$.

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.*

The proof of Theorem 4 follows by carefully showing that the attack constructed in the proof of Theorem 2 can be "approximately" implemented in polynomial-time, if one-way functions do not exists.

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.*

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). 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 ping-pong protocol.

[3] [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.

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 1-bit ping-pong 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 $\eta(n) \geq O(1/\log n)$, 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.

1.2 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.

1.3 Overview of the Paper

In Section 2 we provide some notation and preliminaries. In Section 3 we formally define the notion of knowledge-preserving interactive coding. Section 4 contains our results for the information-theoretic setting, and Section 5 contains our result for the computational setting; finally, in Section 6 we present our impossibility results for non-constructive interactive coding.

2 Notation and Preliminaries

2.1 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.

2.2 Statistical Distance and Computational Indistinguishability

We recall the definitions of *statistical distance* and *computational indistinguishability*.

Definition 7 (Statistical distance)**.** The *statistical distance* between two probability distributions X, Y is defined by $\Delta(X, Y) = (1/2) \cdot \sum_x |\Pr[x \leftarrow X] - \Pr[x \leftarrow Y]|$. X and Y are ϵ-close if $\Delta(X, Y) \leq \epsilon$.

The *statistical distance* between two ensembles $\{X_k\}_k$ and $\{Y_k\}_k$ is a function δ defined by $\delta(k) = \Delta(X_k, Y_k)$. Two probability ensembles are said to be *statistically close* if their statistical distance is negligible. We also say X_k and Y_k are statistically close if $\Delta(X_k, Y_k) \leq \epsilon(k)$ for some negligible function ϵ.

Definition 8 (Computational Indistinguishability)**.** Two ensembles $\{X_k\}, \{Y_k\}$ are *computationally indistinguishable* if for every probabilistic polynomial time distinguisher D, there exists a negligible function μ such that for every $k \in \mathbb{N}$,

$$|\Pr[D(1^k, X_k) = 1] - \Pr[D(1^k, Y_k) = 1]| \leq \mu(k).$$

2.3 Hash Functions

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

Definition 9 (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 $h \leftarrow H$.

2.4 One-way Functions and Distributionally One-way Functions

We start by recalling the definition of a *one-way function*.

Definition 10 (One-way functions)**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if it is computable in polynomial time and for every non-uniform probabilistic polynomial time machine A, there exists a negligible function $\mu(\cdot)$ such that for every $n \in N$,

$$\Pr\left[x \leftarrow \{0,1\}^n : A(f(x)) \in f^{-1}(f(x))\right] < \mu(|x|)$$

Additionally, if there exists some ϵ such that above holds for every $\text{poly}(2^{n^\epsilon})$-sized circuits A, f is a *subexponentially-hard one-way function*.

A *distributionally one way function* is weaker primitive: here it is only computationally infeasible to find a *random* pre-image to $f(x)$ (but finding *some* pre-image may be easy).

Definition 11 ([IL89]:Distributionally one-way functions)**.** We say a function $f : \{0,1\}^* \to \{0,1\}^*$ is *distributionally one-way* if it is computable in polynomial time and there exists a constant $c > 0$ such that for every non-uniform probabilistic polynomial-time algorithm A, for sufficiently large $n \in \mathbb{N}$, the statistical distance between the following distributions is at least $\frac{1}{n^c}$:

- $\{x \leftarrow \{0,1\}^n : (f(x), x)\}$
- $\{x \leftarrow \{0,1\}^n : (f(x), A(f(x_n))\}$

While distributionally one-way functions are a weaker primitive than one way functions, [IL89] shows that the existence of distributionally one-way functions implies the existence of one-way functions.

Theorem 12 ([IL89])**.** *Distributionally one-way functions exist if and only if one way functions exist.*

2.5 Signature schemes

We recall the definition of an (adaptive-secure) signature schemes.

Definition 13 (Signature scheme [GMR89])**.** A *secure signature scheme* with *signature-length* $l(\cdot)$ and *key-length* $v(\cdot)$ is a triple $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ of probabilistic polynomial time algorithms, such that

- for all $n \in \mathbb{N}, m \in \{0,1\}^*$,

$$\Pr[(sk, vk) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sig}_{sk}(m) : |\sigma| \leq l(n) \wedge |vk| \leq v(n) \wedge \mathsf{Ver}_{vk}(m, \sigma) = 1] = 1$$

- for every non-uniform probabilistic polynomial time adversary A, there exists a negligible function $\mu(\cdot)$ such that

$$\Pr[(sk, vk) \leftarrow \mathsf{Gen}(1^n), (m, \sigma) \leftarrow A^{\mathsf{Sig}_{sk}(\cdot)}(1^n) : \mathsf{Ver}_{vk}(m, \sigma) = 1 \wedge m \notin L] \leq \mu(n)$$

  where L denotes the list of A's queries to its oracle.

The existence of signatures schemes is implied by the existence of one-way functions [NY89, Rom90]:

Theorem 14. *Assume the existence of one-way functions (resp. sub-exponentially hard one-way functions). Then there exists a secure signature scheme (resp. a secure signature scheme with signature-length and key-length* polylog$n$*).*

2.6 Error-correcting Codes

We recall the definition of error correcting codes.

Definition 15 (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 16 (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 17 ([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 18 ([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 19. 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 20. *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.*

3 Knowledge-Preserving Interactive Coding

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

3.1 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 21 (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 22 (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 23. 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)[4] channel C that flips at most $\eta(n, m(n)) \cdot CC^*_n(\pi)$ bits, the following holds:

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

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

3.2 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 24. 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.
  - $\{\mathsf{out}[\tilde{A}(x) \leftrightarrow \tilde{B}(y)]\}_{n\in\mathbb{N}, x\in\mathcal{X}_A(1^n), y\in\mathcal{X}_B(1^n)}$
  - $\{\mathsf{out}[A(x) \leftrightarrow B(y)]\}_{n\in\mathbb{N}, x\in\mathcal{X}_A(1^n), y\in\mathcal{X}_B(1^n)}$
- *(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.
  - $\{\mathsf{out}[\tilde{A}^*(x, z) \leftrightarrow \tilde{B}(y)]\}_{n\in\mathbb{N}, x\in\mathcal{X}_A(1^n), y\in\mathcal{X}_B(1^n), z\in\{0,1\}^*}$
  - $\{\mathsf{out}[A^*(x, z) \leftrightarrow B(y)]\}_{n\in\mathbb{N}, x\in\mathcal{X}_A(1^n), y\in\mathcal{X}_B(1^n), z\in\{0,1\}^*}$

  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.

Single-session v.s. Multiple session Knowledge Preservance Our notion of knowledge preservance assumes that the attacker is only participating in a single execution of π', and stipulates that this execution of π' emulates π. A stronger notion of *concurrent* knowledge preservance (in analogy with the notion of concurrent zero-knowledge [DNS04]) would instead require that multiple concurrent executions of π' still emulate π' (in the sense that an attacker participating in an arbitrary polynomial number of sessions of π' can be emulated by a simulator participating in concurrent sessions of π.) We omit a formal definition of concurrent knowledge preservance, and simply remark that our positive results extend also to the concurrent multi-session setting.

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. (Additionally, if π is a concurrently secure implementation of $\mathcal{F}$ and π' is a concurrent knowledge preserving variant of π, then π' is a concurrently secure implementation of $\mathcal{F}$.)

3.3 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 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 25 (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 $\eta'(n,m) = \eta(n)$.

4 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 26. *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 (see Theorem 17); 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 27. *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 exist a knowledge-preserving interactive coding scheme with error rate $\eta(n) = 1/\mathrm{poly}(n)$.)*

Proof. Consider some knowledge-preserving interactive coding protocol $Q = (Q_1, Q_2)$ with information rate $R(\cdot,\cdot)$ and error rate $\eta(\cdot,\cdot)$ 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, ℓ is the communication complexity of the protocol π to be encoded and m is the number of round in π). Assume for contradiction that there exists a polynomial $m(\cdot)$ such that for infinitely many n, $\eta(n, m(n)) \geq 1/m(n)$. We will construct a "*ping-pong*" protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ with round complexity $m(\cdot)$ such that Q^π cannot be a knowledge-preserving variant of π.

The ping-pong protocol π. Let $t(n) = M(n, m(n), 2m(n)n)+1$, and $\mathcal{H}_n = \{H_k : \cup_{i\in\{0,\ldots,2m(n)n\}}\{0,1\}^i \to \{0,1\}^n\}$ be a t-wise independent hash function family. On security parameter n, let $\mathcal{X}_A(1^n) = \mathcal{X}_B(1^n) = \mathcal{H}_n$, i.e., the inputs $x \in \mathcal{X}_A(1^n)$ and $y \in \mathcal{X}_B(1^n)$ for both players specify hash functions H_x and H_y in $\mathcal{H}_n$. π is a deterministic protocol that on the inputs $x \in \mathcal{X}_A(1^n)$ and $y \in \mathcal{X}_B(1^n)$ proceeds as follows: First, A sends $a_1 = H_x(\emptyset)$ to B, who sends back $b_1 = H_y(a_1)$ to A. Then at each round i, A sends $a_i = H_x(a_1, b_1, \ldots, a_{i-1}, b_{i-1})$ to B, who sends back $b_i = H_y(a_1, b_1, \ldots, a_{i-1}, b_{i-1}, a_i)$ to A; namely, both parties generates their next messages by applying their hash function to the current transcript. At the end of the interaction, both A and B output the whole transcript $(a_1, b_1, \ldots, a_m, b_m)$.

Since π is deterministic, the transcript $(a_1, b_1, \ldots, a_m, b_m)$ is determined by the inputs x and y. Let $a_i(x, y)$ (and $b_i(x, y)$ resp.) denote the i-th messages A (and B resp.) send in the interaction of $A(x) \leftrightarrow B(x)$, and $\hat{a}_i(x, y) = (a_1, b_1, \ldots, a_{i-1}, b_{i-1}, a_i)$ and $\hat{b}_i = (a_1, b_1, \ldots, a_i, b_i)$ denote the partial transcripts of the interaction of $A(x) \leftrightarrow B(x)$ up until round i; by definition, $b_i(x, y) = H_y(\hat{a}_i(x, y))$ for $i \in [m]$ and $a_i(x, y) = H_x(\hat{b}_{i-1}(x, y))$ for $i \in [m]$, where $\hat{b}_0(x, y)$ is defined to be $\emptyset$. Let $m'(\cdot)$ denote the round complexity of the encoded protocol Q^π.

Privacy of the Ping-pong Protocol Our first observation is that in the ping-pong protocol, by the unpredictability of the output of the hash functions, player 1, even if maliciously deviating from the protocol instructions, can guess at most m distinct pairs $(q, H_y(q))$, except with negligible probability. Let the predicate $\mathsf{PrivacyBreach}(o, y) = 1$ iff o contains $m + 1$ distinct pairs $(q, H_y(q))$.

Claim 28. *For every adversarial strategy A^* and every $n \in \mathbb{N}$,*

$$\Pr\left[x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n) : \mathsf{PrivacyBreach}(\mathsf{out}_1[A^*(x) \leftrightarrow B(y)], y) = 1\right] \leq L/2^n,$$

where L denotes the length of A^*'s output (i.e., $L = |\mathsf{out}_1[A^*(x) \leftrightarrow B(y)]|$).

Proof. Note that the interaction with $B(y)$ only allows A^* to make m queries to H_y. Thus, for the predicate $\mathsf{PrivacyBreach}$ to output 1, A^* needs to predict one extra pair $(q', H_y(q'))$ that A^* does not learn from B; that is, q' is different from any partial transcript of the interaction. However, since H_y is t-wise independent, $H_y(q')$ remains uniformly random for every such q. Therefore, each guess of A^* in its output can only be correct with probability 2^{-n}. Since A^* can only make at most L guesses in its output, by an union bound, $\Pr[\mathsf{PrivacyBreach}(\mathsf{out}_1[A^*(x) \leftrightarrow B(y)], y) = 1] \leq L/2^n$. □

We shall now see that in the encoded protocol Q^π, this "privacy-property" no longer holds, and as such Q^π cannot be a knowledge preserving variant of π. As a first step towards showing this, we demonstrate a structural property of the encoded protocol Q^π. In the sequel of the proof, we assume without loss of generality that Q never makes the same query twice to its oracle (since the oracle anyway is deterministic).

Implicit Ping-pong Computation. We show that (with overwhelming probability) the encoded protocol Q^π "implicitly executes" the original ping-pong protocol in a chronological order. More precisely, as formalized below, the rounds of the encoded protocol can be divided into (non-empty) "chunks", where each chunk in the encoded protocol corresponds to a single round (i.e., two consecutive messages) in ping-pong protocol; additionally by observing the oracle queries made by Q, we can read out a polynomial list of candidates for the current transcript of the (implicitly executed) ping-pong protocol. We emphasize here that definition of the chunks may depend on the inputs of the players and the randomness of Q.

Formally, let the predicate $\mathsf{ImplicitComp}(x, y, r_1, r_2) = 1$ iff Q_1 asks its oracle (that is, $A(x)$) the queries $\hat{b}_0(x, y)$[5], $\hat{b}_1(x, y), \hat{b}_2(x, y), \ldots, \hat{b}_{m-1}(x, y)$ *in order* during the interaction of $Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_2)$ and all m queries are made in different rounds. When $\mathsf{ImplicitComp}(x, y, r_1, r_2) = 1$, Q_1's queries partition the rounds of the interaction into non-empty chunks in the natural way: for every $i \in [m]$, the *i-th chunk* of the interaction $Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_1)$ starts at the round where Q_1 makes the query $\hat{b}_{i-1}(x, y)$ and finishes when makes Q_1 the query $\hat{b}_i(x, y)$.[6] The following lemma shows that with overwhelming probability (over random inputs x, y and the execution of $Q_1^{A(x)} \leftrightarrow Q_2^{B(y)}$) $\mathsf{ImplicitComp}$ holds and thus chunks are well-defined.

Lemma 29 (Implicit Computation Lemma)**.** *There exists a negligible function $\mu(\cdot)$ such that for every $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n), r_1, r_2 \in \{0,1\}^\infty : \mathsf{ImplicitComp}(x, y, r_1, r_2) = 1] \geq 1 - \mu(n)$$

Intuitively, the lemma follows by the "elusiveness" property of the output of the hash function H_x used by A: if Q_1 is not asking its oracle all the queries in order it must be able to guess the output of H_x on a new point q, which contradicts its t-wise independent property. Note that we here rely on the fact that the output of the hash functions are "long" (i.e., super-logarithmic); otherwise, it is easy to guess the output of the hash function with inverse polynomial probability. We proceed to a formal proof.

Proof (of Lemma 29). For convenience, we actually prove a slightly stronger statement: We show that with overwhelming probability, the following three properties holds during the execution of

[5]Recall that $\hat{b}_0(x, y)$ is defined to be $\emptyset$.

[6]We can assume without loss of generality that Q_1 always queries the full transcript $\hat{b}_m(x, y)$ before generating the output (at the cost of at most one extra query), so that the last chunk m also is well defined.

$Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_2)$: (a) Q_1 makes the queries $\hat{b}_0(x,y), \hat{b}_1(x,y), \hat{b}_2(x,y), \ldots, \hat{b}_{m-1}(x,y)$ to $A(x)$, (b) Q_2 makes the queries $\hat{a}_1(x,y), \hat{a}_2(x,y), \ldots, \hat{a}_m(x,y)$ to $B(y)$, and (c) the queries are made in chronological order; that is, every $i \in [m]$, Q_1 make the query $\hat{b}_{i-1}(x,y)$ before Q_2 makes the query $\hat{a}_i(x,y)$ and Q_2 make the query $\hat{a}_i(x,y)$ before Q_1 makes the query $\hat{b}_i(x,y)$. It easily follows that if the above three properties hold with respect to (x, y, r_1, r_2) then $\mathsf{ImplicitComp}(x, y, r_1, r_2) = 1$. We now show that except with negligible probability (over x, y, r_1, r_2), each of these properties (individually) hold; we can then conclude by a union bound that with overwhelming probability, all of them simultaneously hold.

For (a), suppose for the sake of contradiction that with some noticeable probability $\epsilon(n)$, Q_1 does *not* make the query $\hat{b}_i(x,y)$ for some $i \in \{0, \ldots, m-1\}$. By completeness of Q^π, Q_1 outputs $\hat{b}_m(x,y)$ with overwhelming probability. Thus, by an union bound, with noticeable probability $\epsilon'(n)$, Q_1 does not query $\hat{b}_i(x,y)$ but outputs $\hat{b}_m(x,y)$, which contains $a_i(x,y) = H_x(\hat{b}_i(x,y))$. But, this can only happen with probability at most 2^{-n} since H_x is t-wise independent and Q_1 makes at most $t-1$ queries to its oracle, none of which is $\hat{b}_i(x,y)$, and its oracle is exactly computing $H_x(\cdot)$; thus, even conditioned on Q_1 view of the interaction, $H_x(\hat{b}_i(x,y))$ is uniform and can thus only be guessed by Q_1 with probability 2^{-n}. By identically the same argument it follows that (b) happens except with negligible probability.

For (c), suppose for the sake of contradiction that with some noticeable probability $\epsilon(n)$, Q_2 make the query $\hat{a}_i(x,y)$ before Q_1 makes the query $\hat{b}_{i-1}(x,y)$ for some $i \in [m]$. Note that $\hat{a}_i(x,y)$ contains $a_i(x,y) = H_x(\hat{b}_{i-1}(x,y))$. Thus, by guessing which query of Q_2 is $\hat{b}_{i-1}(x,y)$, we can construct an algorithm R that predicts the value of $H_x(\hat{b}_{i-1}(x,y))$ with probability $\epsilon(n)/t(n)$, while querying H_x on at most $t(n)-1$ points, none of which is $\hat{b}_{i-1}(x,y)$, contradicting t-wise independence of H_x. By identically the same argument if follows that Q_1 only make the query $\hat{b}_i(x,y)$ before Q_2 makes the query $\hat{a}_i(x,y)$ for some $i \in [m]$ with negligible probability. □

Obtaining a Privacy Breach in the Encoded Protocol We are now ready show how to obtain a privacy breach in the encoded protocol Q^π.

Claim 30. *There exists an adversarial strategy $\tilde{A}^*$ for player 1 in Q^π such that for sufficiently large $n \in \mathbb{N}$, the output length of $\tilde{A}^*$ is bounded by $2M(n,m,2mn)m(n)n$ and*

$$\Pr[x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n) : \mathsf{PrivacyBreach}(\mathsf{out}_1[\tilde{A}^*(x) \leftrightarrow Q_2^{B(y)}], y) = 1] \geq 1/4m'(n)$$

Proof. The idea behind the attacker $\tilde{A}^*$ (acting as player 1) is the following. By an averaging argument, one of the chunks, say chunk i, in the encoded protocol must be shorter than a fraction $1/m$ of the total communication complexity of the encoded protocol. The idea now is for $\tilde{A}^*$ to honestly execute the encoded protocol using its actual input x and randomness r_1, except that during the i'th chunk, $\tilde{A}^*$ samples a fresh input x' (and a fresh randomness r_1' for Q_1) consistent with the transcript up until (and including) chunk $i-1$,[7] and executes the chunk using the input x' (and randomness r_1') instead of x. Intuitively, since the chunk was "small", by the error resilience property of the interactive coding scheme, player 1 will finally produce the same output (including m pairs $(q, H_y(q))$) as if it had been running the protocol honestly (that is, without "deviating" in chunk i). Formally proving this requires showing that the attack performed by $\tilde{A}^*$ can be modelled as channel that flips a sufficiently small number of bits; indeed, note that the way $\tilde{A}^*$ deviates from the protocol does not rely the original random coins r_1 used by Q_1 or the input x, and this property is crucial for us to be able to emulate the attacker by a channel.

[7] Note that this step may not be efficiently computable in general, but this is not a problem since we here consider information-theoretic security.

Finally, by observing the oracle queries made by Q_1 during the i'th chunk—which corresponds to the i'th round in the ping-pong protocol, by the "implicit computation" property—$\tilde{A}^*$ may learn an additional pair $(q', H_y(q'))$; intuitively, the reason for this is that, with overwhelming probability, the messages in the i-th round in "implicit computation" of π remain uniformly distributed, even after conditioning on the first $i-1$ rounds. (We, however, warn the reader that proving this is quite subtle; see Sub-claim 32). Thus, if $\tilde{A}^*$ could just identify the i'th chunk, it can learn $m+1$ pairs $(q, H_y(q))$ of H_y. Towards this, $\tilde{A}^*$ simply guesses the starting round of the i'th chunk.

We proceed to a formal description of the attacker $\tilde{A}^*$. (Recall that $m'(\cdot)$ denotes the round complexity of Q^π.) On input $x \in \mathcal{X}_A(1^n)$, and randomness r_1, $\tilde{A}^*$ performs the follow "forking" attack:

1. $\tilde{A}^*$ uniformly picks a random round $j \leftarrow [m'(n)]$ and honestly executes the encoded protocol Q_1 up to the end of $(j-1)$-th round. Let T be the resulting partial transcript.
2. $\tilde{A}^*$ samples a fresh input-randomness pair (x', r_1') conditioned on the partial transcript T; namely, (x', r_1') are uniformly random over all input-randomness pair such that $Q_1^{A(x')}(r_1')$ is consistent with T.[8] Then, $\tilde{A}^*$ continues executing Q_1 but now with inputs x' and randomness r_1', for as many rounds as possible, subject to the restriction that the number of bits it transmitted since round j does not exceed $\eta(n,m) \cdot \mathrm{CC}_n(Q^\pi)$.
3. $\tilde{A}^*$ continues the rest of the interaction honestly, with the "true" input x and randomness r_1 of Q_1 pretending that its own messages were honestly sent all along (including the messages sent since round j), but may have been incorrectly received by player 2. At the end of the interaction, $\tilde{A}^*$ outputs (o, L), where o is the output of Q_1, and L is the list of queries Q_1 made during the "deviation" (using the new input x').

We first show that, with overwhelming probability, $\tilde{A}^*$ can still learn the "valid" m pairs $(q, H_y(q))$ (that it would have learn even if it didn't deviate).

Sub-claim 31. *There exist a negligible function $\mu(\cdot)$ such that for every $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n), (o, L) \leftarrow \mathsf{out}_1[\tilde{A}^*(x) \leftrightarrow Q_2^{B(y)}] : o = \mathsf{out}_1[A(x) \leftrightarrow B(y)]] \geq 1 - \mu(n).$$

Proof. Note that the deviation performed by $\tilde{A}^*$ in Step 2 can be implemented by a channel C, since it only relies on knowledge of the transcript of the interaction and not the internal state (inputs and randomness) of either of the players. Also, by construction of $\tilde{A}^*$, C never needs to flip more than $\eta \cdot \mathrm{CC}_n(Q^\pi)$ bits. The claim follows directly by the η-error resilience and completeness of Q^π. □

Note that $o = \mathsf{out}_1[A(x) \leftrightarrow B(y)]$ contains m distinct pairs $(q, H_y(q)$, namely, $(\hat{a}_i(x,y), b_i(x,y) = H_y(\hat{a}_i(x,y))$ for $i \in [m]$; below we abuse of notation and let o denote the set of of these pairs. We next show that L contains one additional distinct pair $(q', H_y(q'))$ with noticeable probability. Recall that a query of Q_1 is of the form $(a_1, b_1, \ldots, a_i, b_i)$; below we sometimes abuse of notation and interpret each such query as a pair (q, v) where $q = (a_1, b_1, \ldots, a_i)$ and $v = b_i$ (that is, q is the vector containing all but the last component, and v contains only the last component).

Sub-claim 32. *For sufficiently large $n \in \mathbb{N}$, the following holds*

$$\Pr\left[\begin{array}{l} x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n), (o, L) \leftarrow \mathsf{out}_1[\tilde{A}^*(x) \leftrightarrow Q_2^{B(y)}] : \\ \exists (q, H_y(q)) \in L \text{ and } (q, H_y(q)) \notin \mathsf{out}_1[A(x) \leftrightarrow B(y)] \end{array}\right] \geq 1/2m'(n).$$

[8] Note that this is the only inefficient step in the forking attack.

Proof. Note that in the experiment $\{x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n) : \mathsf{out}_1[\tilde{A}^*(x) \leftrightarrow Q_2^{B(y)}]\}$, for every choice of j (by $\tilde{A}^*$), the tuple (x', y, r_1', r_2) is uniformly distributed (since we can view the experiment as first sampling a uniform $j-1$-round transcript T and then then sampling (x', y, r_1', r_2) conditioned on T). It follows that the distribution of (x', y, r_1', r_2) is independent of j.

Additionally, note that $\tilde{A}^*$ could have picked x', r_1' is a somewhat more convoluted way, generating exactly the same distribution: instead of directly sampling x', r_1' conditioned on T, first sample a list L_1 of (at most $t(n)-1$) query-answer pairs corresponding to Q_1's queries to its oracle $A(x)$ up until round $j-1$, conditioned on the transcript being T; next, sample x' conditioned on L_1 and T, and finally r' conditioned on x', L_1 and T. The following observation will be useful in what follows:

> *Sampling x' conditioned on L_1 and T is equivalent to sampling x' just conditioned on just the query-answer pairs L_1.*

The reason this holds is that conditioned on L_1, x' and T are independent (recall that T is determined as a function of just L_1, y, r_1, r_2).

Now, by observing the list of queries L_1 (up to the transcript T) and the input y of player 2, let us determine the next round in the "implicit computation" of π (or said otherwise, the "next chunk" in the encoded protocol after the transcript T). If L_1 does not contains the query $\emptyset$, let $i = 1$ (i.e., the implicit computation has not begun yet since player 1 has not generated its first input a_1; consequently, the next round is 1). If L_1 contains the query-answer pair $(\emptyset, a_1)$ (since Q_1's oracle A is deterministic, there can be at most one such query), check if L_1 contains a query of the form $(b_1 = H_y(a_1), a_2)$ (again, there can be at most one such query); if not set $i = 2$, otherwise, check if L_1 contains a query of the form $(b_2 = H_y(a_1, b_1, a_2), a_3)$, and so on.

Below we show the following two statements:

1. With probability at least $1/m'(n) - \mathrm{negl}(n)$, it holds that $i \in [m(n)]$ (i.e., the implicit computation of π has not ended) and L contains the query $\hat{b}_i(x', y)$ (which corresponds to the pair $(\hat{a}_i(x', y), b_i(x', y) = H_y(\hat{a}_i(x', y)))$
2. With probability at most 2^{-n}, it holds that $(\hat{a}_i(x', y), b_i(x', y)) \in \mathsf{out}_1[A(x) \leftrightarrow B(y)]$.[9]

The claim then follows by a union bound.

To prove statement (1), consider some fixed (x', y, r_1', r_2) where chunks are well-defined (i.e. $\mathsf{ImplicitComp}(x', y, r_1', r_2) = 1$). Note that the event in (1) means that the whole i-th chunk of $Q_1^{A(x')}(r_1') \leftrightarrow Q_2^{B(y)}(r_2)$ is completed during the deviation, i.e., the chunk starts at or after round j, and ends before the "cut-off". By an averaging argument, there must exist some chunk i^* that is shorter than $(1/m) \cdot \mathrm{CC}_n(Q^\pi) \leq \eta(n, m) \cdot \mathrm{CC}_n(Q^\pi)$. Clearly, when j equals to the starting round of chunk i^*, which happens with probability $1/m'(n)$ since j is uniformly distributed and independent of (x', y, r_1', r_2), chunk $i = i^*$ and thus chunk i will be completed before the cut-off; consequently, since $\mathsf{ImplicitComp}(x', y, r_1', r_2) = 1$, L contains the query $\hat{b}_i(x', y)$. Since, by Lemma 29 (and the observation that x', y, r_1', r_2 are uniformly distributed) $\mathsf{ImplicitComp}(x', y, r_1', r_2) = 1$ holds with overwhelming probability, we have $\Pr[\hat{b}_i(x', y) \in L] \geq 1/m'(n) - \mathrm{negl}(n)$ for some negligible function $\mathrm{negl}(n)$.

To prove statement (2), note that if $a_i(x, y) \neq a_i(x', y)$, then $(\hat{a}_i(x', y), b_i(x', y)) \notin \mathsf{out}_1[A(x) \leftrightarrow B(y)]$. Thus, it sufficient to show that $\Pr[a_i(x, y) = a_i(x', y)] \leq 2^{-n}$; that is $\Pr[H_x(\hat{b}_{i-1}(x, y)) = H_{x'}(\hat{b}_{i-1}(x', y))] \leq 2^{-n}$. Consider some fixed x, y, r_1, r_2, L_1; note that i is determined as a function of these (recall that it is a function of L_1, y), and trivially $H_x(\hat{b}_{i-1}(x, y)$ is determined. Additionally,

[9] Pedantically, in case $i > m(n)$, $\hat{a}_i$ is not defined; in this case the event is simply defined to not hold.

note that for every x' consistent with L_1, $\hat{b}_{i-1}(x', y) = H_y(\hat{a}_{i-1}(x', y)$ is determined and furthermore $\hat{b}_{i-1}(x', y)$ is not contained in L_1 (since by definition of i, $\hat{b}_{i-2}(x', y)$ is the "last" ping-pong query in L_1). Since, as observed above, x' is uniformly sampled conditioned only on L_1 (and since L_1 contains at most $t(n) - 1$ queries, none of which is $\hat{b}_{i-1}(x', y)$), it follows that *every* fixed x, y, r_1, r_2, L_1, $\Pr[H_x(\hat{b}_{i-1}(x, y) = H_{x'}(\hat{b}_{i-1}(x', y)] \leq 2^{-n}$, and thus this condition also holds for random x, y, r_1, r_2.

□

By combining Sub-claim 31 and 32, and applying the union bound we get that

$$\Pr[\mathsf{PrivacyBreach}(\mathsf{out}_1[\tilde{A}^*(x) \leftrightarrow Q_2^{B(y)}], y) = 1] \geq 1/2m'(n).$$

which concludes the proof of Claim 30. □

Combining Claim 28 and 30 shows that Q^π is not a knowledge-preserving variant of π, which leads to a contradiction and completes the proof of Theorem 27. □

5 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).

5.1 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 33. *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* $1/\text{polylog}n$.

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 messages 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.[10] 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.

We provide a formal description of the scheme in Figure 1. We assume without loss of generality that in the original protocol π each player sends a *single bit* at each round in the protocol (we refer to such protocols as "bit protocols").

The following key lemma shows that the above-described scheme is a knowledge preserving interactive coding schemes. The proof of Theorem 33 is then concluded by appropriately setting ρ and c.

Lemma 34. *Let $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ be a bit protocol with round complexity $m(\cdot)$. Let $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a secure signature scheme with signature length $l(n)$ and verification-key length $v(n)$, let $(E_{\mathsf{p}}, D_{\mathsf{p}})$ be an ECC with constant error rate η_{p} and information rate $R_p(\cdot)$ and $(E_{\mathsf{m}}, D_{\mathsf{m}})$ be an error correcting codes that is efficiently η_{m}-list-decodable and has information rates $R_p(\cdot)$. Consider Q, α, β, γ defined in Figure 1 w.r.t the functions $\rho(\cdot,\cdot), c(\cdot,\cdot)$. If $c(n,m) \geq \gamma(n,m)$, then Q is a computational knowledge preserving interactive coding scheme with perfect completeness, and:*

- *information rate $R(\cdot)$ s.t.*
$$R(n,m) = c(n,m)/2m;$$
- *error rate $\eta(\cdot,\cdot)$ s.t.*
$$\eta(n,m) = \min\left(\eta_{\mathsf{m}} \cdot \frac{c(n,m) - \gamma(n,m)}{2c(n,m)}, \eta_{\mathsf{p}} \cdot \frac{\alpha(n)}{c(n,m)}\right).$$

Proof. We separately prove each of the required properties.

[10] [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.

Input protocol: A bit protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ with round complexity $m = m(n)$.

Parameters:

- Let $\rho = \rho(n, m)$ be a *padding* parameter.
- Let $c = c(n, m)$ be a *cut-off* parameter.

Primitives used:

- Let $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme with signature length $l = l(n)$ and verification key length $v = v(n)$.
- Let $(E_{\mathsf{p}}, D_{\mathsf{p}})$ be an error correcting code with (constant) error rate η_{p} and information rate $R_p = R_p(\cdot)$.
- Let $(E_{\mathsf{m}}, D_{\mathsf{m}})$ be an error correcting code that is efficiently η_{m}-list decodable and has information rate $R_m = R_m(\cdot)$.

Initialization: On input a security parameter 1^n, round complexity 1^m, and communication complexity 1^{2m}, Q_1 (resp., Q_2) initializes a counter $i_1 = 1$ (resp., $i_2 = 1$).

Preamble: Q_1 runs $(sk_1, vk_1) \leftarrow \mathsf{Gen}(1^n)$ and sends $c_{1,0} = E_{\mathsf{p}}(vk_1||0^\rho)$ to Q_2. Then Q_2 runs $(sk_2, vk_2) \leftarrow \mathsf{Gen}(1^n)$ and sends $c_{2,0} = E_{\mathsf{p}}(vk_2||0^\rho)$ to Q_1. Both Q_1 and Q_2 decode the received message $c'_{2,0}$ and $c'_{1,0}$ and store $vk'_2 = D_{\mathsf{p}}(c'_{2,0})$ and $vk'_1 = D_{\mathsf{p}}(c'_{1,0})$, respectively. Let $\alpha(n)$ denote the length of each message $c'_{1,0}$, $c'_{2,0}$ in the preamble phase; that is $\alpha(n) = (v(n) + \rho(n))/R_p(v(n) + \rho(n))$

Main: We first define the strategy of Q_1:

- First, Q_1 queries its oracle A to obtain the first message a_1, appends a_1 to t_1, computes $\sigma_1 \leftarrow \mathsf{Sig}_{sk_1}((i_1, a_1))$, sends $c_{1,1} = E_{\mathsf{m}}((i_1, a_1, \sigma_1))$ to Q_2, and increases the counter by 1 (i.e., $i_1 := i_1 + 1$).
- Upon receiving a message c' from Q_2, Q_1 list-decodes c, and verifies that there exists a *unique* message (i', b', σ') such that (a) $i' = i_1 - 1$ and (b) $\mathsf{Ver}_{vk'_2}((i', b'), \sigma') = 1$. If the verification rejects, Q_1 resends its previous message c_{1,i_1-1}. If the verification accepts, then Q_1 appends b' to t_1 and queries t_1 to its oracle A. If A returns an output o_1, then Q_1 outputs o_1 and terminates. If A returns a next message a_{i_1}, then Q_1 appends a_{i_1} to t_1, computes $\sigma_{i_1} \leftarrow \mathsf{Sig}_{sk_1}((i_1, a_{i_1}))$, sends $c_{1,i_1} = E_{\mathsf{m}}((i_1, a_{i_1}, \sigma_{i_1}))$ to Q_2, and increases the counter by 1 (i.e., $i_1 := i_1 + 1$).

The strategy of Q_2 is defined analogously, except that in step (a) , Q_2 verifies that $i' = i_2$ (as opposed to $i_2 - 1$).

- Upon receiving a message c' from Q_1, Q_2 list-decodes c, and verifies that there exists a *unique* message (i', b', σ') such that (a) $i' = i_2$ and (b) $\mathsf{Ver}_{vk'_1}((i', a'), \sigma') = 1$. If the verification rejects, Q_2 resends its previous message c_{2,i_2-1}. If the verification accepts, then Q_2 appends a' to t_2 and queries t_2 to its oracle B. If B returns an output o_2, then Q_2 outputs o_2 and terminates. If B returns a next message b_{i_2}, then Q_2 appends b_{i_2} to t_2, computes $\sigma_{i_2} \leftarrow \mathsf{Sig}_{sk_2}((i_2, a_{i_2}))$, sends $c_{2,i_2} = E_{\mathsf{m}}((i_2, b_{i_2}, \sigma_{i_2}))$ to Q_1, and increases the counter by 1 (i.e., $i_2 := i_2 + 1$).

Let $\beta(n)$ denote the length of each round in the main phase. Let $\gamma(n, m) = 2\alpha(n) + 2m\beta(n)$.

Abort condition: At any point, if sending the next message causes the total communication to exceed $c(n, m)$ bits, the scheme Q aborts (with the players outputting $\perp$).

Figure 1: Interactive coding scheme $Q = (Q_1, Q_2)$ encoding an interactive bit protocol π.

Efficiency and Information rate: Q is clearly polynomial-time computable, and by definition of Q, we have that $CC^*(Q^\pi) = c(n,m)$; it follows that Q has information rate $R(n,m) = c(n,m)/(2m)$.

Perfect Completeness: It easily follow from the definition of Q that Q^π perfectly emulates π if both players are honest. In fact, for every n, input pair $x \in \mathcal{X}_A(1^n), y \in calX_B(1^n)$ and randomness pair $r_A, r_B \in \{0,1\}^\infty$,

$$\Pr\left[\mathsf{out}[Q_1^{A_{r_A}(x)} \leftrightarrow Q_2^{B_{r_B}(y)}] = \mathsf{out}[A_{r_A}(x) \leftrightarrow B_{r_B}(y)]\right] = 1$$

We refer to this property as *perfect completeness*, and it will be useful shortly.

Knowledge Preservance: First note that if $c(n,m) \geq \gamma(n,m)$ then Q is complete. Note that Q executes π in a straight-line fashion (without any "rewindings"). We now use this observation to show that $\tilde{\pi} = Q^\pi$ is a knowledge preserving variant of π. More precisely, for every polynomial-time attacker $\tilde{A}^*$ for player 1, we show the existence of a polynomial-time simulator A^* such that for every $x \in \mathcal{X}_A$, $y \in \mathcal{X}_B$ and $z \in \{0,1\}^*$ we have that $\mathsf{out}_{\tilde{A}^*}[\tilde{A}^*(x,z) \leftrightarrow Q_2^{B(y)}]$ is identically distributed to $\mathsf{out}_{A^*}[A^*(x,z) \leftrightarrow B(y)]$. The simulator $A^*(x,z)$, in essence, is just the encoding algorithm Q_2: $A^*(x,z)$ simply emulates an interaction between $\tilde{A}^*(x,z)$ and Q_2, while externally forwarding all the oracle queries by Q_2 and answering those queries by forwarding back all the external message. Since Q_2 never rewinds its oracle, the view of $\tilde{A}^*$ in the simulation is identical to its view in the real execution. (Note that the knowledge preservance property holds unconditionally.) A simulator for an adversarial player 2 is constructed in the analogous way.

Error resilience We turn to showing that Q^π is η-error-resilient, where

$$\eta(n) = \min\left(\eta_{\mathsf{m}} \cdot \frac{c(n,m(n)) - \gamma(n,m(m))}{2c(n,m(n))}, \eta_{\mathsf{p}} \cdot \frac{\alpha(n)}{c(n,m(n))}\right)$$

Consider some non-uniform polynomial-time channel C, security parameter n, inputs x, y and randomness r_A, r_B and an execution $e \in supp(Q_1^{A_{r_A}(x)} \leftrightarrow_{C(1^n)} Q_2^{B_{r_B}(y)})$ such that $\mathsf{out}(e) \neq \mathsf{out}(A_{r_A}(x) \leftrightarrow B_{r_B}(y))$ (recall that by perfect completeness, for every $e' \in supp(Q_1^{A_{r_A}(x)} \leftrightarrow Q_2^{B_{r_B}(y)})$, we have that $\mathsf{out}(e) = \mathsf{out}(A_{r_A}(x) \leftrightarrow B_{r_B}(y))$.) For this to happen, either of two things must have happened:

1. either execution gets cut-off (in which case both players output $\perp$); or,
2. either Q_1 or Q_2 queries its oracle on a partial transcript t' that is not a partial transcript in the execution of $A(x) \leftrightarrow B(y)$.

We show that neither of these cases can happen with inverse polynomial probability for infinitely many n (and selections of x, y, r_A, r_B) as long as C corrupts at most $\eta(n)c(n,m(n))$ bits.

Let us first prove that case 1 only can happen with negligible probability. First, note that since $c(n,m(n)) \geq \gamma(n,m(n))$ we have that Q^π does not abort when run over a noiseless channel. Next, note that since $\eta(n) \leq \eta_{\mathsf{p}}\alpha(n)/c(n,m(n))$, the channel can corrupt at most $\eta_{\mathsf{p}} \cdot \alpha(n)$ bits. It follows that each message in the preamble phase will always be correctly decoded, since $(E_{\mathsf{p}}, D_{\mathsf{p}})$ has error rate η_{p} and each message in the preamble phase is of length $\alpha(n)$.

Additionally, since $\eta(n) \leq \eta_{\mathsf{m}} \cdot (c(n, m(n)) - \gamma(n, m(n)))/(2c(n, m(n))$, the channel can corrupt at most

$$\eta_{\mathsf{m}} \cdot \frac{(c(n, m(n)) - \gamma(n, m(n)))}{2}$$

bits. Note that unless channel corrupts $\eta_{\mathsf{m}}\beta(n)$ bits in a message in the main phase, the message can still be list decoded. Furthermore, it follows (as in [MPSW10]), relying on the security of the signature scheme (against non-uniform polynomial-time attackers)[11] that except with negligible probability, the correct message is the unique one that has an accepting signature (since the channel has seen at most a single signed message of the form $(i, \cdot)$ for each verification key). On the other hand, if the channel corrupts more than $\eta_{\mathsf{m}}\beta(n)$ bits in one message, list decoding is no longer guaranteed to work, and this may cause the players to resend *two* messages (recall that the player that notices a corrupted message, resends its previously send message, which forces the other player to resend the corrupted one). Thus, every time the channel corrupts $\eta_{\mathsf{m}}\beta(n)$ bits, it may cause the interaction to become $2\beta(n)$ bits longer. So, to make the interaction become cut-off (with non-negligible probability), we need j such corruptions of $\eta_{\mathsf{m}}\beta(n)$ bits, where

$$\gamma(n) + 2\beta(n)j > c(n, m(n)).$$

In other words,

$$j > \frac{c(n, m(n)) - \gamma(n, m(n))}{2\beta(n)},$$

which means that the channels needs to corrupt more than

$$\eta_{\mathsf{m}} \cdot \frac{c(n, m) - \gamma(n, m)}{2}$$

bits, which is a contradiction.

We proceed to show that case 2 only can happen with negligible probability. The key observation needed to prove this is that, as mentioned above, the preamble phase is always uniquely decoded and thus C cannot change the verification keys vk_1, vk_2. Consider the first time that, say, Q_1 queries its oracle on a partial transcript t' that is not a partial transcript in the execution of $A(x) \leftrightarrow B(y)$. It means that Q_1 accepts an incorrect message (i', b', σ') such that b' is different from the message b_{i_1-1} it should have received in π in round $i_1 - 1$, it must be the case that a) $i' = i_1 - 1$ (or else Q_1 would reject it), and b) C provided a valid signature (for the verification key vk_2) on (i', b'). But the channel has seen at most a single signature (for the verification key vk_2) on a message of the type $(i_1 - 1, \cdot)$ (since Q_2 will only send a single message of this type); it thus follows from the non-uniform security of the signature scheme that player 2 accepts an incorrect message with at most negligible probability. It follows analogously that C can only make player 2 accept an incorrect message with negligible probability. □

Equipped with Lemma 34, we now turn to proving Theorem 33.

Proof of Theorem 33. Assume the existence of one-way functions, fix some $\epsilon > 0$. By scaling down the security parameter in the construction from Theorem 14, there exists a signature scheme with both verification-key length and signature length $O(n^\epsilon)$. Additionally, by Theorem 18 and 20, there exists ECCs $(E_{\mathsf{p}}, D_{\mathsf{p}})$, $(E_{\mathsf{m}}, D_{\mathsf{m}})$ with information rate $R(n) = O(1/n)$, and such that $(E_{\mathsf{p}}, D_{\mathsf{p}})$ has

[11] Note that the reason we require non-uniform security of the signature scheme is that the attacker needs to get the inputs x, y and the non-uniform advice of C.

error-rate $1/4 - \epsilon$ and $(E_{\mathsf{m}}, D_{\mathsf{m}})$ is $1/2 - \epsilon$ efficiently list decodable. By Lemma 34, we get that the error rate $\eta(n)$ is (approximately) maximized[12] when

$$c(n,m) = \gamma(n,m) + \alpha(n) = 3\alpha(n) + 2m(n)\beta(n).$$

(that is, the two expressions inside the *min* are the same). In this case,

$$\eta(n) = \eta_{\mathsf{p}} \cdot \frac{\alpha(n)}{3\alpha(n) + 2m(n)\beta(n)}.$$

Note that we can set the padding parameter $\rho(n)$ to be a sufficiently big polynomial such that $\epsilon \cdot \alpha(n) \geq 2m(n)\beta(n)$; it follows that the error rate is at least

$$\frac{\eta_{\mathsf{p}}}{3+\epsilon} = \frac{(1/4) - \epsilon}{3+\epsilon} \geq \frac{1}{12} - \epsilon$$

By Lemma 34, the information rate of Q is $c(n,m)/2m = O(n^{\epsilon})$. This concludes the first part of the theorem.

If additionally assuming the existence of subexponentially-hard one-way functions, it instead follows that $c(n,m) \leq O(m \cdot \text{polylog}n)$. □

5.2 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 35. *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 forking attacker $\tilde{A}^*$ in the proof of Theorem 27 can be approximated efficiently if one-way functions do not exist, and so can the channel adversary. We turn to a formal proof. Assume for contradiction that one-way functions do not exist, we show that there does not exist a computational knowledge-preserving interactive coding scheme with polynomial error and information rate.

Consider some polynomial $m(\cdot)$ and computational knowledge-preserving interactive coding protocol $Q = (Q_1, Q_2)$ with error rate $\eta(n,m) \geq 1/m(n)$; 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 and ℓ are the round complexity and communication complexity of the protocol π to be encoded).

Let $t = M(n,m,2mn) + 1$. Let $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$ be the ping-pong protocol defined in the proof of Theorem 27 with m rounds and using t-wise independence hash functions as inputs. Let $m'(\cdot)$ be the round complexity of the encoded protocol Q^{π}. We show that Q^{π} cannot be a computational knowledge-preserving variant of π by obtaining a privacy breach using the same

[12] For simplicity, we ignore ϵ terms.

forking attacker $\tilde{A}^*$ as in the proof of Theorem 27, but relying on the assumed non-existence of one-way functions, to make $\tilde{A}^*$ and the channel C that emulates the attacker $\tilde{A}^*$, efficient.

Recall that the only inefficient step of $\tilde{A}^*$ is in Step 2, where $\tilde{A}^*$ needs to sample a fresh input-randomness pair (x', r'_1) conditioned on the first $j-1$ rounds partial transcript T of the execution $Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_2)$. Let f denote the function that maps (x, y, r_1, r_2, j) to T. Clearly, f is efficient. By Theorem 12 and the assumed non-existence of one-way functions, there exists a polynomial-time inverter M such that $(T, M(T))$ and $(T, (x, y, r_1, r_2, j))$ has statistical distance at most $1/8m'(n)$ for infinitely many $n \in \mathbb{N}$. (Note that M only works for infinitely many $n \in \mathbb{N}$. Nevertheless, this is sufficient.) Namely, M can approximately sample a random pre-image of a random transcript T with small inverse polynomial error. Relying on the inverter M, $\tilde{A}^*$ can be made efficient straightforwardly, with the following modification; let us denote the modified attacker $\tilde{A}^*_{\mathsf{eff}}$.

- In the second step, $\tilde{A}^*_{\mathsf{eff}}$ applies M to the partial transcript T to obtain a pre-image $(x', y'r'_1, r'_2, j')$. Then $\tilde{A}^*_{\mathsf{eff}}$ uses (x', r'_1) to continue the attack as before. Namely, $\tilde{A}^*_{\mathsf{eff}}$ continues the interaction, but with the input and randomness to Q_1 switched to x' and r'_1, for as many rounds as possible, subject to that the number of bits it transmits since round j does not exceed $\eta(n) \cdot \mathrm{CC}_n(Q^\pi)$.

By construction, the attacker $\tilde{A}^*_{\mathsf{eff}}$ runs in polynomial time; additionally, since the channel C used in the proof of Theorem 27 perfectly mimics the attacker, it is also efficient.

We now show that $\tilde{A}^*_{\mathsf{eff}}$ approximate $\tilde{A}^*$ well over random inputs, which implies that $\tilde{A}^*_{\mathsf{eff}}$ can also obtain a privacy breach with inverse polynomial probability for infinitely many n. Specifically, we show that the following two experiments have a statistical distance of at most $1/8m'(n)$ for infinitely many $n \in \mathbb{N}$.

- $\mathrm{Exp}_1(1^n) = \left\{ x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n) : \tilde{A}^*(x) \leftrightarrow Q_2^{B(y)} \right\}$
- $\mathrm{Exp}_2(1^n) = \left\{ x \leftarrow \mathcal{X}_A(1^n), y \leftarrow \mathcal{X}_B(1^n) : \tilde{A}^*_{\mathsf{eff}}(x) \leftrightarrow Q_2^{B(y)} \right\}$

Claim 36. *For infinitely many $n \in \mathbb{N}$, the statistical distance between $\mathrm{Exp}_1(1^n)$ and $\mathrm{Exp}_2(1^n)$ is at most $1/8m'(n)$.*

Proof. Recall that both experiments are determined by the values of $(x, y, r_1, r_2, j, x', r'_1)$, where (x, y, r_1, r_2, j) are independently and uniformly sampled, and then in Exp_1, (x', r'_1) are sampled conditioned on the first $j-1$ round partial transcript T of the execution $Q_1^{A(x)}(r_1) \leftrightarrow Q_2^{B(y)}(r_2)$, and in Exp_2, (x', r'_1) are sampled by first applying the efficient inverter to obtain (x', y', r'_1, r'_2, j') and then discarding (y', r'_2, j').

Fix a security parameter $n \in \mathbb{N}$ such that M works; that is, such that $(T, M(T))$ and $(T, (x, y, r_1, r_2, j))$ have a statistical distance of at most $1/8m'(n)$. This implies that the distributions of (T, x', r'_1) in $\mathrm{Exp}_1(1^n)$ and $\mathrm{Exp}_2(1^n)$ have a statistical distance of at most $1/8m'(n)$, since projection (i.e., removing (y', r'_2, j')) cannot increase statistical distance. We claim that additionally, the distribution of the whole tuple $(x, y, r_1, r_2, j, x', r'_1)$ in $\mathrm{Exp}_1(1^n)$ and $\mathrm{Exp}_2(1^n)$ have a statistical distance of at most $1/8m'(n)$ as well. To see this, as a thought experiment, we can view the experiments as sampled in the following different order: first, T is sampled, then (x', r'_1) are sampled conditioned on T, and finally (x, y, r_1, r_2, j) are sampled conditioned on T (note that conditioned on T, (x', r'_1) and (x, y, r_1, r_2, j) are independent). Note that in both experiments, (x, y, r_1, r_2, j) are sampled in an identical way after sampling (T, x', r'_1), so it does not increase the statistical distance. Therefore, the statistical distance between $\mathrm{Exp}_1(1^n)$ and $\mathrm{Exp}_2(1^n)$ on this security parameter n is at most $1/8m'(n)$. □

Now, since $\tilde{A}^*$ obtains a privacy breach with probability $\geq 1/4m'(n)$ in $\text{Exp}_1(1^n)$ for sufficiently large $n \in \mathbb{N}$, and $\text{Exp}_1(1^n)$ and $\text{Exp}_2(1^n)$ have statistical distance at most $1/8m'(n)$ for infinitely many $n \in \mathbb{N}$, it follows that $\tilde{A}^*_{\mathsf{eff}}$ can also obtains a privacy breach with probability $\geq 1/8m'(n)$ in $\text{Exp}_2(1^n)$ for infinitely many $n \in \mathbb{N}$. This shows that Q^π is not a computationally knowledge-preserving variant of π and completes the proof. □

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 33 (assuming subexponentially hard one-way functions) is essentially optimal.

Theorem 37. *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. The theorem is a direct consequence of Theorem 38 proven in Section 6. □

6 Lower Bound for Non-constructive Schemes

In this section, we present an impossibility result for the computational setting, which applies even to *non-constructive* interactive coding scheme. In particular, for every $\eta(n) > 1/\log n$, we demonstrate 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 27, we here provide an impossibility result also for non-constructive interactive coding.)

Theorem 38. *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 $\text{CC}_n(\pi) = O(1/\eta(n))$ 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 $\omega(\log n)$.*

Proof. We here consider a somewhat 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). We show that *any* protocol π' with communication complexity $O(\log n)$ that is computational η-error resilient cannot be a computationally knowledge preserving variant of π.

Similarly to the proof of Theorem 27, let us first formally define π and formalize a security property that it satisfies.

The "bit-exchange" protocol π. Let $m(\cdot) = \lceil 2/\eta(\cdot) \rceil$ and consider the following simple bit-exchange protocol $\pi = (A, B, \mathcal{X}_A, \mathcal{X}_B)$, where both players simply send their inputs to each other, bit-by-bit. On security parameter n, let $m = m(n)$ and $\mathcal{X}_A(1^n) = \mathcal{X}_B(1^n) = \{0,1\}^{m(n)}$. π is a deterministic m-round protocol that on the inputs $x \in \{0,1\}^m$ and $y \in \{0,1\}^m$ proceeds as follows: at each round $i \in [m]$, A sends $a_i = x_i$ to B, who sends back $b_i = y_i$ to A, where x_i, y_i denote the i-th bit of x, y, respectively. At the end of the interaction, both A and B output the whole transcript (a, b), where $a = (a_1, \ldots, a_m) \in \{0,1\}^m$ and $b = (b_1, \ldots, b_m) \in \{0,1\}^m$.

"Guess-the-next-bit" security of the bit-exchange protocol. We say that player *matches* in a round $i \in [m]$ if it guesses the bit sent by its opponent in the next message. Formally, for $i \in [m]$, defined the predicates $\mathsf{Match}_{1,i}(a,b) = 1$ iff $a_i = b_i$, $\mathsf{Match}_{2,i}(a,b) = 1$ iff $b_i = a_{i+1}$. Note that if the inputs are uniformly distributed, then for every $i \in [m]$, the probability that each player matches in round i with probability *exactly* $1/2$.

Claim 39. *For every adversarial strategy A^*, every $n \in \mathbb{N}$ and $i \in [m]$,*

$$\Pr\left[x, y \leftarrow \{0,1\}^m : \mathsf{Match}_{1,i}(\mathsf{out}_2[A^*(x) \leftrightarrow B(y)]) = 1\right] = 1/2.$$

Similarly, for every adversarial strategy B^*, every $n \in \mathbb{N}$ and $i \in [m-1]$,

$$\Pr\left[x, y \leftarrow \{0,1\}^m : \mathsf{Match}_{2,i}(\mathsf{out}_1[A(x) \leftrightarrow B^*(y)]) = 1\right] = 1/2.$$

Proof. The claim trivially follows by noting that for every A^* (resp., B^*), y_i (resp., x_{i+1}) remains uniformly random conditioned on the view of A^* (resp., B^*) when A^* (resp., B^*) needs to send its i-th message. □

Now, consider some computationally knowledge preserving variant $\pi' = (Q_1, Q_2, \{0,1\}^{m(n)}, \{0,1\}^{m(n)})$ of π that has $O(\log n)$ communication complexity and is η-error resilient. Let $c(n) = \max\{\mathrm{CC}_n(\pi'), \log n\}$, and $m'(\cdot)$ be the round complexity of π'. As in the proof of Theorem 27, we first show that π' needs to be "implicitly executing" π in a chronological order. In contrast to this step in the proof of Theorem 27, we here rely on the knowledge preservance property of π' to demonstrate this.

Implicit bit-exchange computation. Our formalization of implicit computation here is quite different from how it was formalized in the proof of Theorem 27. Here we rely on information-theoretic definitions of what it means to implicitly execute π (which is what allows us to provide an impossibility result also for non-constructive coding schemes). Towards formalizing it, let us first introduce some additional definitions.

For two strings T and T', we denote by $T \preceq T'$ (resp., $T \prec T'$) that T is a prefix (resp., proper prefix) of T'. Fix a security parameter n and let $m = m(n)$. If $x, y \in \{0,1\}^m$ and T is a partial transcript, then let $p(x,y,T) = \Pr[T \preceq \mathsf{trans}[Q_1(x) \leftrightarrow Q_2(y)]]$; that is, the probability that T is consistent with the execution of $Q_1(x) \leftrightarrow Q_2(y)$. Let $\delta > \epsilon \in (0,1)$ be two parameters (to be determined later). For inputs $x, y \in \{0,1\}^m$, a partial transcript T, $i \in [m]$ and $\beta \in \{0,1\}$, we say that (y,T) (resp., (x,T)) is (δ,ϵ)-binding for (i,β) if the following two conditions hold:

1. There exists $x' \in \{0,1\}^m$ with $x'_i = \beta$ (resp., $y' \in \{0,1\}^m$ with $y'_i = \beta$) such that $p(x',y,T) \geq \delta$ (resp., $p(x,y',T) \geq \delta$).
2. For every $x' \in \{0,1\}^m$ with $x'_i \neq \beta$ (resp., $y' \in \{0,1\}^m$ with $y'_i \neq \beta$), $p(x',y,T) \leq \epsilon$ (resp., $p(x,y',T) \leq \epsilon$).

Intuitively, this means that (y,T) "determines" $x_i = \beta$. Consider an execution $T \leftarrow \mathsf{trans}[Q_1(x,r_1) \leftrightarrow Q_2(y,r_2)]$ for some inputs $x, y \in \{0,1\}^m$ and randomness $r_1, r_2 \in \{0,1\}^\infty$. Note that since the probability $p(x,y,T)$ can be estimated by sampling, the above two conditions can be checked in time $\mathrm{poly}(2^c, 1/\delta, 1/\epsilon)$, which allows player 1 to "decode" the bit y_i from (x,T) when (x,T) is binging for player 2's i-th bit input. Specifically, there is an algorithm Dec_1 with runtime $\mathrm{poly}(n, 2^c, 1/\delta, 1/\epsilon)$ that takes (x,T,i,δ,ϵ) as input such that (a) if (x,T) is (δ,ϵ)-binding for (i,β), then Dec_1 outputs β with probability at least $1 - 2^{-n}$, and (b) if (x,T) is *not* $(\delta/2, 2\epsilon)$-binding for both $(i,0)$ and $(i,1)$, then Dec outputs $\perp$ with probability at least $1 - 2^{-n}$. Analogously, there is an algorithm Dec_2 to "decode" player 1's bit x_i from (y,T).

Define the *i-th binding-point* for player 1 (resp., 2) of the execution (x, y, r_1, r_2) with parameters (δ, ϵ) to be the shortest partial transcript $T_{1,i} \preceq T$ (resp., $T_{2,i} \preceq T$) such that (i) the last message in $T_{1,i}$ (resp., $T_{2,i}$) is sent by player 1 (resp. player 2), and (ii) $(y, T_{1,i})$ (resp., $(x, T_{2,i})$) is (δ, ϵ)-binding for (i, x_i) (resp., (i, y_i)); if no such partial transcript exists, define $T_{1,i} = \perp$ (resp., $T_{1,i} = \perp$). For convenient, we use notation $T_{d,i}(x, y, r_1, r_2; \delta, \epsilon)$ to denote the i-th binding-point for player d of the execution (x, y, r_1, r_2) with parameters (δ, ϵ).

Intuitively, $T_{d,i}$ is the point where player d sends its i-bit to the other player (corresponding to the i-th message in the implicit computation of π). Formally, let the predicate $\mathsf{ImplicitComp}(x, y, r_1, r_2; \delta, \epsilon) = 1$ iff (i) $T_{d,i}(x, y, r_1, r_2; \delta, \epsilon) \neq \perp$ for every $d \in \{1, 2\}$ and $i \in [m]$, (ii) $T_{1,i} \prec T_{2,i}$ for every $i \in [m]$, and (iii) $T_{2,i} \prec T_{1,i+1}$ for every $i \in [m-1]$; that is, the i-th binding-points for both players occur in a chronological order corresponding to the execution of π and do not occur at the same time. When $\mathsf{ImplicitComp}(x, y, r_1, r_2; \delta, \epsilon) = 1$, the interaction of π' can be partitioned into non-empty chunks as before: for every $i \in [m]$, the *i-th chunk* of the execution $Q_1(x, r_1) \leftrightarrow Q_2(y, r_1)$ starts after $T_{2,i-1}$ and finishes at the end of $T_{2,i}$; that is, the i-th chunk starts when player 1 starts sending i-message and finished when player 1 receives the i-th message from player 2.

The following lemma shows that $\mathsf{ImplicitComp}$ holds with high probability for every input pair (x, y) and appropriate (sufficiently small) inverse polynomial δ and ϵ.

Lemma 40 (Implicit Computation Lemma)**.** *For every polynomial $q(n) \geq n^5 \cdot 2^{5(m(n)+c(n))}$, for sufficiently large $n \in \mathbb{N}$,*

$$\Pr[x, y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \in \{0,1\}^\infty : \mathsf{ImplicitComp}(x, y, r_1, r_2; 1/q(n), 1/q^2(n)) = 1] \geq 1 - 1/n.$$

Proof. Let $\delta(n) = 1/q(n)$ and $\epsilon(n) = 1/q^2(n)$. Let $\mu(n)$ be a negligible function such that π' is a computational knowledge preserving variant of π with respect to $\mu(n)$ (for both completeness and computational knowledge preservance). Recall that the completeness property says that for every $n \in \mathbb{N}$, every $x, y \in \{0,1\}^{m(n)}$,

$$\Pr[\mathsf{out}[Q_1(x) \leftrightarrow Q_2(y)] = \mathsf{out}[A(x) \leftrightarrow B(y)] \geq 1 - \mu(n).$$

Fix n to be a sufficiently large security parameter such that $\mu(n) \leq 1/q^5(n)$. We first show that for every $x, y \in \{0,1\}^m$,

$$\Pr[r_1, r_2 \leftarrow \{0,1\}^\infty : \forall d \in \{1,2\}, i \in [m], T_{d,i}(x, y, r_1, r_2; \delta, \epsilon) \neq \perp] \geq 1 - 1/2n.$$

Namely, condition (i) of the predicate $\mathsf{ImplicitComp}$ is satisfied for every inputs $x, y\{0,1\}^{m(n)}$ with high probability. Note that it suffices to show that with probability at least $1 - 1/2n$ over $T \leftarrow \mathsf{trans}[Q_1(x) \leftrightarrow Q_2(y)]$, for every $i \in [m]$, (y, T) is (δ, ϵ)-binding for (i, x_i) and (x, T) is (δ, ϵ)-binding for (i, y_i); that is, every bit of both players' input are eventually binding at the end of the execution. To show this, it suffices to show the following two statements.

1. For every $x, y \in \{0,1\}^{m(n)}$, $\Pr[T \leftarrow \mathsf{trans}[Q_1(x) \leftrightarrow Q_2(y)] : p(x, y, T) \geq \delta(n)] \geq 1 - 1/2n$.
2. For every $x \neq x' \in \{0,1\}^{m(n)}$, $y \neq y' \in \{0,1\}^{m(n)}$, and a *full* transcript $T \in \{0,1\}^*$,

$$\min\{p(x, y, T), p(x', y, T)\} \leq 3\mu(n) \text{ and } \min\{p(x, y, T), p(x, y', T)\} \leq 3\mu(n).$$

Statement (1) implies that condition (1) in the definition of binding is satisfied with probability at least $1 - 1/2n$, and when the event in statement (1) holds, statement (2) guarantees that condition (2) is also satisfied. Now, statement (1) follows by noting that

$$\Pr[T \leftarrow \mathsf{trans}[Q_1(x) \leftrightarrow Q_2(y)] : p(x, y, T) \leq \delta(n)] \leq \delta(n) \cdot 2^{c(n)} \leq 1/2n.$$

For statement (2), suppose that there exist $x \neq x' \{0,1\}^{m(n)}$, $y \in \{0,1\}^{m(n)}$ and a full transcript $T \in \{0,1\}^*$ such that both $p(x,y,T), p(x',y,T) \geq 3\mu(n)$. Consider two executions $Q_1(x) \leftrightarrow Q_2(y)$ and $Q_1(x') \leftrightarrow Q_2(y)$. Note that conditioned on the full transcript, the output of player 2 is independent of the input of player 1. Thus, when the transcript is T, player 2 must generate an incorrect output for one of the two executions (i.e., $\mathsf{out}_2[Q_1(x) \leftrightarrow Q_2(y)] \neq \mathsf{out}_2[A(x) \leftrightarrow B(y)]$, which is simply (x,y)). Since in both executions, T occurs with probability at least $3\mu(n)$, it follows that player 2 produces incorrect output in one of the execution with probability $\geq 3\mu(n)/2 > \mu(n)$, violating the completeness property. This shows that $\min\{p(x,y,T), p(x',y,T)\} \leq 3\mu(n)$ for every $x \neq x' \in \{0,1\}^m(n)$, $y \in \{0,1\}^m(n)$, and full transcript $T \in \{0,1\}^*$. An analogous argument shows that $\min\{p(x,y,T), p(x,y',T)\} \leq 3\mu(n)$ for every $x \neq x' \in \{0,1\}^m(n)$, $y \in \{0,1\}^m(n)$, and full transcript $T \in \{0,1\}^*$.

We proceed to show that condition (ii) and (iii) hold with high probability, relying on the following claim.

Claim 41. *For every $i \in [m]$, suppose one of the following holds.*

- $\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : x_i \neq y_i \wedge T_{2,i}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{1,i}(x,y,r_1,r_2;\delta,\epsilon)] \geq 1/q(n)$, *or*
- $\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : x_i = y_i \wedge T_{2,i}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{1,i}(x,y,r_1,r_2;\delta,\epsilon)] \geq 1/q(n)$.

Then there exists an adversarial strategy $\tilde{A}^*$ for player 1 such that

$$\left|\Pr\left[x,y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : \mathsf{Match}_{1,i}(o) = 1\right] - 1/2\right| \geq 1/8q(n).$$

Similarly, for every $i \in [m-1]$, suppose one of the following holds.

- $\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : y_i \neq x_{i+1} \wedge T_{1,i+1}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{2,i}(x,y,r_1,r_2;\delta,\epsilon)] \geq 1/q(n)$, *or*
- $\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : y_i = x_{i+1} \wedge T_{1,i+1}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{2,i}(x,y,r_1,r_2;\delta,\epsilon)] \geq 1/q(n)$, *or*

Then there exists an adversarial strategy $\tilde{B}^*$ for player 2 such that

$$\left|\Pr\left[x,y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[Q_1(x) \leftrightarrow \tilde{B}^*(y)] : \mathsf{Match}_{2,i}(o) = 1\right] - 1/2\right| \geq 1/8q(n).$$

Note that the conclusions of the claim contradict to the knowledge preserving property. Thus, it follows that for every $i \in [m]$,

$$\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : T_{2,i}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{1,i}(x,y,r_1,r_2;\delta,\epsilon)] \leq 2/q(n), \textit{and}$$

$$\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : T_{1,i+1}(x,y,r_1,r_2;\delta,\epsilon) \prec T_{2,i}(x,y,r_1,r_2;\delta,\epsilon)] \leq 2/q(n).$$

It now follows by a union bound that condition (ii) and (iii) hold with probability at least $1 - 4m/q(n) \geq 1 - 1/2n$, and another union bound concludes that

$$\Pr[x,y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \in \{0,1\}^\infty : \mathsf{ImplicitComp}(x,y,r_1,r_2;1/q(n),1/q^2(n)) = 1] \geq 1 - 1/n.$$

It remains to prove the claim.

Proof of Claim 41. We start by proving the first case of the claim. Namely, we show that if

$$\Pr[x, y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : x_i \neq y_i \wedge T_{2,i}(x, y, r_1, r_2; \delta, \epsilon) \prec T_{1,i}(x, y, r_1, r_2; \delta, \epsilon)] \geq 1/q(n),$$

then

$$\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : \mathsf{Match}_{1,i}(o) = 1\right] \geq 1/2 + 1/8q(n).$$

At a high level, the idea is simple: whenever y_i is bound at point $T_{2,i}$ before x_i is bound, $\tilde{A}^*$ can first decode y_i from $(x, T_{2,i})$, and then if $x_i = y_i$, $\tilde{A}^*$ simply continues honestly, but if $x_i \neq x_i$, $\tilde{A}^*$ changes its input from x to some x' with $x'_i = y_i$ ($\tilde{A}^*$ can do so since x_i is not bound yet). Formally, on input $x \in \{0,1\}^m$, $\tilde{A}^*$ performs the following strategy to interact with Q_2:

- $\tilde{A}^*$ samples a uniform randomness $r_1 \leftarrow \{0,1\}^\infty$ and starts by honestly executing the protocol Q_1, but at the end of each round j, $\tilde{A}^*$ runs $\mathrm{Dec}_1(x, T_j, i, \delta, \epsilon)$ where T_j is the current partial transcript. If Dec_1 outputs $\perp$, then $\tilde{A}^*$ simply continues honestly at round $j+1$. If Dec_1 outputs $\beta \in \{0,1\}$, let T^* denote the current transcript and $\tilde{A}^*$ proceeds as follows.
  - If $\beta = x_i$, then $\tilde{A}^*$ simply continues the execution honestly throughout.
  - If $\beta \neq x_i$, then $\tilde{A}^*$ uses rejection sampling to sample a random (x', r'_1) conditioned on $x'_i = \beta$ and the current transcript T^*. $\tilde{A}^*$ cuts-off the rejection sampling procedure when it fails for more than $M = O(n2^m/\delta\epsilon)$ samples; in this case, $\tilde{A}^*$ sets $(x', r'_1) = (x, r_1)$ (i.e., does not change the input-randomness pair). Then $\tilde{A}^*$ continues executing the protocol Q_1 with the alternative input-randomness pair (x', r'_1) throughout.

We proceed to analyze $\tilde{A}^*$. The following sub-claim says that $\mathsf{Match}_{1,i}$ holds with high probability when $x_i = y_i$.

Sub-claim 42. $\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : x_i = y_i \wedge \mathsf{Match}_{1,i}(o) = 1\right] \geq 1/2 - 1/8q(n)$.

Proof. First note that by completeness, we have

$$\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[Q_1(x) \leftrightarrow Q_2(y)] : x_i = y_i \wedge \mathsf{Match}_{1,i}(o) = 1\right] \geq 1/2 - \mu(n).$$

Additionally, note that when $x_i = y_i$ and there is no "decoding error" (i.e., during the execution Dec_1 does not return $\beta = \bar{y}_i$), then $\tilde{A}^*$ simply executes Q_1 honestly. Thus, to lower bound the probability of the desired event, it suffices to upper bound the probability that a decoding error occur during the execution. Now, observe that during the execution, the property of Dec_1 ensures that when $p(x, y, T) > 2\epsilon$, $\mathrm{Dec}_1(x, T, i, \delta, \epsilon)$ outputs incorrect answer $\beta = \bar{y}_i$ with probability at most 2^{-n}. Noting that for every (x, y), $\Pr[T \leftarrow \mathsf{trans}[Q_1(x) \leftrightarrow Q_2(y)] : p(x, y, T) \leq 2\epsilon] \leq 2^{c(n)} \cdot 2\epsilon$, and that $\tilde{A}^*(x)$ invokes Dec_1 at most $m'(n)$ times, the probability that a decoding error occur can be upper bounded by $m'(n) \cdot 2^{-n} + 2^{c(n)} \cdot 2\epsilon$. A final union bound implies that

$$\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : x_i = y_i \wedge \mathsf{Match}_{1,i}(o) = 1\right]$$

$$1/2 - \mu(n) - m'(n) \cdot 2^{-n} - 2^{c(n)} \cdot 2\epsilon \geq 1/2 - 1/8q(n).$$

□

We next show that when $x_i \neq y_i$, $\mathsf{Match}_{1,i}$ still holds with noticeable probability.

Sub-claim 43. $\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : x_i \neq y_i \wedge \mathsf{Match}_{1,i}(o) = 1\right] \geq 1/4q(n)$.

Proof. Let Good denote the event that during the execution, the tuple (x, y, T^*) satisfies the following three conditions: (i) $x_i \neq y_i$, (ii) $p(x, y, T^*) \geq \delta$, (iii) $\mathrm{Dec}_1(x, T^*, i, \delta, \epsilon)$ outputs y_i (i.e., Dec_1 decodes correctly), and (iv) let T^- denote the partial transcript obtained by removing the last message from T^* (thus, the last message is from player 1 to player 2); (y, T^-) is *not* (δ, ϵ)-binding for (i, x_i).

We first show that Good happens with probability at least $1/2q(n)$. Recall that $\tilde{A}^*$ executes Q_1 honestly before the end of T^*, and that in the execution of $Q_1(x) \leftrightarrow Q_2(y)$,

$$\Pr[x, y \leftarrow \{0,1\}^{m(n)}, r_1, r_2 \leftarrow \{0,1\}^\infty : x_i \neq y_i \wedge T_{2,i}(x, y, r_1, r_2; \delta, \epsilon) \prec T_{1,i}(x, y, r_1, r_2; \delta, \epsilon)] \geq 1/q(n).$$

Now, consider any tuple (x, y, r_1, r_2) such that the above event hold, and let $T_{2,i} = T_{2,i}(x, y, r_1, r_2; \delta, \epsilon)$ and $T^-_{2,i}$ be $T_{2,i}$ with the last message removed. Note that $\mathrm{Dec}_1(x, T_{2,i}, i, \delta, \epsilon)$ outputs y_i with probability at least $1 - 2^{-n}$, and that $(y, T^-_{2,i})$ is *not* (δ, ϵ)-binding for (i, x_i). Therefore, in the execution of $\tilde{A}^*(x) \leftrightarrow Q_2(y)$, conditioned on such (x, y, r_1, r_2), we have that with probability at least $1 - m'(n) \cdot 2^{-n}$, it holds that the above four conditions hold with $T^* \preceq T_{2,i}$ (the $m'(n) \cdot 2^{-n}$ term comes from union bound over at most $m'(n)$ invocation of Dec_1; note that it is possible that $T^* \prec T_{2,i}$ and in this case, the probability that Dec_1 outputs $\bar{y}_i$ is also upper bounded by 2^{-n}). Therefore, the probability that the Good event happens is at least $1/q(n) \cdot (1 - m'(n) \cdot 2^{-n} \geq 1/2q(n)$.

We next show that conditioned on any tuple (x, y, T^*) such that the Good event holds, with probability at least $(1 - 1/8q(n))$, (a) $\tilde{A}^*$ can successfully sample a fresh input-randomness pair (x', r'_1) through rejection sampling, and (b) Q_2 outputs (x', y) at the end of the execution. For (a), the fact that (y, T^-) is not (δ, ϵ)-binding for (i, x_i) and $p(x, y, T^-) \geq p(x, y, T^*) \geq \delta$ implies that there exists some x^* such that $x^*_i \neq x_i$ and $p(x^*, y, T^-) \geq \epsilon$. Now, the facts that the last message of T^* is sent by player 2 and that $p(x, y, T^*) \geq \delta$ imply that $p(x^*, y, T^*) \geq \epsilon\delta$. Therefore, the chance of sampling a pair (x', r'_1) that is consistent with T^* is at least $2^{-m}\epsilon\delta$, and rejection sampling with $M = O(n2^m/\delta\epsilon)$ samples can succeed with probability at least $1 - 2^{-n}$. For (b), note that the execution now is equivalent to an honest execution of $Q_1(x') \leftarrow Q_2(y)$ conditioned on transcript T^*. By completeness, Q_2 outputs (x', y) with probability at least $1 - \mu(n)/p(x', y, T^*)$. Now, recall that there exists x^* with $p(x^*, y, T^*) \geq \epsilon\delta$. A Markov argument shows that the rejection sampling procedure returns an x' with $p(x', y, T^*) \leq \epsilon\delta 2^{-m}/8q(n)$ is at most $1 - 1/8q(n)$. Therefore, when condition (a) holds, we have that condition (b) also holds with probability at least $(1 - 1/8q(n)) \cdot (1 - \mu(n)8q(n)2^m/\epsilon\delta) \geq (1 - 1/4q(n))$. Therefore, the probability that both conditions hold is at least $(1 - 2^{-n}) \cdot (1 - 1/4q(n)) \geq 1 - 1/2q(n)$.

Finally, we can conclude that

$$\begin{aligned}&\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : x_i \neq y_i \wedge \mathsf{Match}_{1,i}(o) = 1\right] \\ &\geq 1/2q(n) \cdot (1 - 1/2q(n)) \geq 1/4q(n).\end{aligned}$$

□

Combining the above two sub-claims, we have that

$$\begin{aligned}&\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*(x) \leftrightarrow Q_2(y)] : \mathsf{Match}_{1,i}(o) = 1\right] \\ &\geq 1/2 - 1/8q(n) + 1/4q(n) \geq 1/2 + 1/8q(n).\end{aligned}$$

This completes the proof of the first case of the claim. The remaining three cases of the claim can be proved by an analogous argument.

□

□

Obtaining a match-deviation in π' Now that we have established that implicit computation holds in π' with high probability, we can obtain a deviation on the matching probability in π' relying on a similar forking attack as in the proof of Theorem 27; such deviation shows that π' cannot be a knowledge-preserving variant of π.

Let $\delta(n) = 1/n^5 \cdot 2^{5(m(n)+c(n))}$ and $\epsilon(n) = \delta^2(n)$. For $i \in [m]$ and $e \in \{0,1\}$, consider the following adversarial strategy $\tilde{A}^*_{i,e}$ for player 1. On input $x \in \{0,1\}^m$, and randomness r_1, $\tilde{A}^*$ performs the follow "forking" attack:

1. $\tilde{A}^*_{i,e}$ uniformly picks a random round $j \leftarrow [m'(n)]$ and honestly executes the encoded protocol Q_1 up to the end of $(j-1)$-th round. Let T be the resulting partial transcript.
2. $\tilde{A}^*_{i,e}$ samples a fresh input-randomness pair (x', r'_1) conditioned on the partial transcript T using rejection sampling with a sufficiently large cuts-off parameter $M = 1/\epsilon^5$. Then, $\tilde{A}^*$ continues executing Q_1 but now with inputs x' and randomness r'_1, for as many rounds as possible, subject to the restriction that the number of bits it transmitted since round j does not exceed $\eta(n) \cdot \mathrm{CC}_n(\pi')$. Let T' be the resulting partial transcript.
3. $\tilde{A}^*_{i,e}$ invokes $\mathrm{Dec}_1(x, T', i, \delta, \epsilon)$, and proceeds with the following two cases.
   - If Dec_1 outputs $\perp$ or $\beta \in \{0,1\}$ such that $\beta \oplus x_i = e$, then $\tilde{A}^*_{i,e}$ continues the rest of the interaction honestly, with the "true" input x and randomness r_1 of Q_1 (pretending that it was player 2 that deviated since round j, but its own messages were correctly sent). (pretending that it sent its own messages correctly but was corrupted by the channel).
   - If Dec_1 outputs $\beta \in \{0,1\}$ such that $\beta \oplus x_i \neq e$, then $\tilde{A}^*_{i,e}$ samples another fresh input-randomness pair (x'', r''_1), conditioned on the partial transcript T *and* that $\beta \oplus x''_i = e$, using rejection sampling with a sufficiently large cuts-off parameter $M = 1/\epsilon^5$; if the rejection sampling fails, $\tilde{A}^*_{i,e}$ sets $(x'', r''_1) = (x, r_1)$ (i.e., $\tilde{A}_{i,e}$ does not change the input-randomness pair). Then $\tilde{A}^*_{i,e}$ continues the rest of the interaction honestly, with the "new" input x'' and randomness r''_1 of Q_1 (again, pretending that it sent its own messages correctly but was corrupted by the channel).
4. $\tilde{A}_{i,e}$ produces no output.

Claim 44. *For sufficiently large n, there exists $i \in [m(n)]$ and $e \in \{0,1\}$ such that*

$$\left|\Pr\left[x, y \leftarrow \{0,1\}^m, o \leftarrow \mathsf{out}_2[\tilde{A}^*_{i,e}(x) \leftrightarrow Q_2(y)] : \mathsf{Match}_{1,i}(o) = 1\right] - 1/2\right| \geq 1/32m'(n).$$

Proof. The claim is proved by combining the analysis of Sub-claim 32 in Theorem 27 and the proof of Claim 41 above.

Recall that implicit computation holds with probability at least $1-1/n$, and in such case, chunks are well-defined. Let $i^* \in [m]$ be such that the i^*-th chunk has shortest expected length when chunks are well-defined. By an averaging argument, the expected length of the i^*-th chunk is at most $(1/m) \cdot \mathrm{CC}_n(\pi')$. By an identical argument to the analysis of Sub-claim 32, for both $e \in \{0,1\}$, in the experiment $\{x, y \leftarrow \{0,1\}^m : \tilde{A}^*_{i^*,e}(x) \leftrightarrow Q_2(y)\}$, (x', y, r'_1, r_2) is uniformly distributed and independent of j. By a Markov argument, with probability at least $1/2$, the i^*-th chunk has length at most $(2/m) \cdot \mathrm{CC}_n(\pi) < \eta(n) \cdot \mathrm{CC}_n(\pi)$. Define Good to be the event that (i) chunks are well-defined, (ii) the i^*-th chunk has length at most $(2/m) \cdot \mathrm{CC}_n(\pi)$ and (iii) j equals to the starting round

of the i^*-th chunk. Note that when Good happens, the i^*-th chunk finished before T' and when $\tilde{A}_{i^*,e}$ invokes Dec_1, it returns a correct bit $\beta = y_i$ with overwhelming probability. Additionally, by definition, (y, T^-) is not (δ, ϵ)-binding for x_i, where T^- is obtained by removing the last message from T. Note that Good happens with probability at least $(1-1/n)\cdot(1/2)\cdot(1/m'(n))\cdot(1-\mathrm{negl}(n)) \geq 1/4m'(n)$.

By a similar argument to the proof of Sub-claim 43, when Good happens, Q_2 outputs (a, b) with $a_i \oplus b_i = e$ with high probability. Roughly, the reason is that in this case, with high probability (a) $\tilde{A}^*_{i^*,e}$ can sample a input-randomness pair (x'', r''_1) such that $x''_i \oplus y_i = e$ and (b) the error-resilient property implies that Q_2 outputs (x'', y). Therefore, for either $e = 0$ or $e = 1$, $\tilde{A}^*_{i^*,e}$ can gain at least $1/16m'(n)$ advantage from the Good event. On the other hand, as argued in the proof of Sub-claim 42, the only chance that $\tilde{A}^*_{i^*,e}$ is when Dec_1 returns incorrect answer $\beta = \bar{y}_i$, which happens with negligible probability. Therefore, the overall advantage of $\tilde{A}^*_{i,e}$ is at last $1/32m'(n)$. □

□

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.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.

[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.

[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.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[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.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC '89*, pages 33–43, 1989.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[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.