Communication vs. Computation

Prahladh Harsha[*] Yuval Ishai [†] Joe Kilian [‡] Kobbi Nissim [§] S. Venkatesh [¶]

April 25, 2004

Abstract

We initiate a study of tradeoffs between communication and computation in well-known communication models and in other related models. The fundamental question we investigate is the following: Is there a computational task that exhibits a strong tradeoff behavior between the amount of communication and the amount of time needed for local computation?

Under various standard assumptions, we exhibit boolean functions that show strong tradeoffs in the following computation models: (1) two-party randomized communication complexity; (2) query complexity; (3) property testing. For the model of deterministic communication complexity, we show a similar result relative to a random oracle.

Finally, we study a time-degree tradeoff problem that arises in arithmetization of boolean functions, and relate it to time-communication tradeoff questions in multi-party communication complexity and in cryptography.

[*]Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA, E-mail: prahladh@mit.edu. Research done while the author was at NEC Laboratories America.

[†]Computer Science Department, Technion, Haifa 32000, Israel, E-mail: yuvali@cs.technion.ac.il.

[‡]NEC Laboratories America Inc, Princeton, NJ 08540, USA, E-mail: joe@nec-labs.com.

[§]Microsoft Research, SVC. 1065 La Avenida, Mountain View, CA 94043, USA, E-mail: kobbi@microsoft.com. Research done while the author was at NEC Laboratories America.

[¶]Computer Science Department, University of Victoria, Victoria, BC, Canada V8W 3P6, E-mail: venkat@cs.uvic.ca. Research done while the author was at MPI for Informatik, Germany.

1 Introduction

A Motivating Riddle. Consider the following multi-party communication game. Fix a finite field F and let M be a $n \times k$ matrix over F. The columns of F are assigned to k players so that each player j knows all columns of M *except* the jth. (This is known as the "input on the forehead" model [CFL83].) The players' goal is to compute the product of the n row sums, namely the function

$$\mathsf{PS}(M) = \prod_{i=1}^{n} \sum_{j=1}^{k} M_{i,j},$$

by means of simultaneously sending messages to an external referee. This can be easily done by having the entire matrix M sent to the referee (e.g., letting P_1 send the second column and P_2 the remaining columns). The goal is to minimize the *communication complexity*, measured as the length of the longest message sent. A closely related problem was studied in [BGKL03]. When $k > n$ (say, $k = n + 1$) our problem admits the following simple solution, implicit in [BGKL03]. Write $\mathsf{PS}(M)$ as the sum of k^n terms, where each term is a product involving a single entry from each row of M. Since there are more players than rows, for each such term there is a player holding all of its values. Hence, one can assign each term to some player who knows its value, and have each player send the sum of all terms assigned to it. The referee can then recover $\mathsf{PS}(M)$ by simply adding up the k field elements it received. While this protocol is very efficient in communication, the combined *computation* of the players is exponential in n. Note that if one uses the natural greedy strategy of assigning each term to the *first* player to which it can be assigned, then player $n + 1$ will need to compute the *permanent* of an $n \times n$ sub-matrix of M, a $\#P$-hard problem.[1] Thus, a natural question is the following:

> Does the function $\mathsf{PS}(M)$ admit a protocol in which (1) each player only sends a single element of F; and (2) the local computation of each player is polynomial in n?

A negative answer seems likely in light of the failure of the natural term assignment strategy. It also seems reasonable that for *any* valid way of assigning the k^n terms to the players, some player will be forced to compute a hard function. Thus, this problem looks like a good candidate for a *time-communication tradeoff*: it requires little time to compute when there is no limit on the communication complexity, requires little communication when there is no limit on the time complexity, but seems to defy solutions that are simultaneously efficient with respect to both complexity measures.

Quite surprisingly, it turns out that the answer to the above question is "yes". (The impatient reader can skip to Section 5.2 for a solution to the riddle.) Thus, this particular problem does not exhibit the time-communication tradeoff that was initially suspected. However, this question served as the original motivation for this work, which explores the existence of similar kinds of tradeoffs in related contexts.

1.1 Problem Description

Let $f : X \times Y \to Z$ be an arbitrary function of two inputs. In the two-party communication model of Yao [Yao79], there are two players A and B. A is given $x \in X$, B is given $y \in Y$ and they

[1] Even if F has characteristic 2, in which case the permanent can be efficiently computed, it is not clear that the computation of (say) the middle player can be made efficient.

need to compute $z = f(x, y)$ by communicating with each other. In any communication protocol designed for f, there are three useful measures of complexity:

- **Communication complexity:** The total number of bits exchanged between A and B;
- **Time complexity:** The amount of time needed by A and B for local computation;
- **Round complexity:** The number of messages exchanged by A and B.

Given any two of these three complexity measures, it is natural to ask if there are tasks which exhibit a tradeoff between them. The question of rounds vs. computation does not arise in the two-party model, as the simple protocol in which A send his entire input over to B is optimal with respect to both measures.[2] Tradeoffs between round complexity and communication complexity have been well studied (see below). In this paper, we initiate the study of the remaining question: proving tradeoffs between communication and local computation. Specifically, our main goal is to find functions f such that: (1) f can be efficiently computed given both its inputs, i.e., given no restriction on the communication; (2) f has a protocol with low communication complexity given no restriction on the computation; and (3) there is no protocol for f which simultaneously has low communication and efficient computation.

To give a simple example in a specific communication model, suppose A is given a CNF formula g as input and B an assignment x to its variables. They need to decide whether x satisfies g by having A send a single message to B. Moreover, suppose that A is deterministic. In this setting, it is easy to argue that there is no protocol that simultaneously uses efficient computation and *optimal* communication, unless $P = NP$. Indeed, such a protocol would allow to efficiently decide equivalence between two given formulas g_1, g_2 by simply comparing the message sent by A on these two inputs. Note that this tradeoff result is very weak, as it does not rule out time-efficient protocols in which the communication is, say, twice larger than the optimal. Our goal is to obtain stronger tradeoff results in various models.

1.2 Related work

Papadimitriou and Sipser [PS84] first discussed the problem of showing tradeoffs between rounds of communication and communication complexity. For any fixed k, they proposed a boolean function p_k called the *pointer chasing problem* that has a k-round protocol with $O(\log n)$ bits of communication. They conjectured that its communication complexity is at least linear if only $k-1$ rounds are allowed. In other words, p_k shows a strong tradeoff behavior between rounds and communication complexity. This conjecture was proved in a series of papers ([PS84, DGS87, NW93]).

Additional complexity measures which are not considered in this work are *space* complexity and *randomness* complexity. Tradeoffs between space and communication were considered by Beame et al. [BTY94]. Tradeoffs between randomness and communication were studied by Canetti and Goldreich [CG93].

1.3 Our Results

Our first result is a strong time-communication tradeoff for a boolean function in the two-party randomized communication model.

[2] However, this question does make sense in a cryptographic setting when players need to compute a function of their inputs without revealing their inputs to each other. Such a tradeoff question is addressed in Section 5.3.

Randomized communication model. Suppose that there is a UP relation R such that the search problem corresponding to R is not in BPTIME$[2^{O(T(n))}]$. (This would follow from the existence of a one-way permutation secure against a $2^{O(T(n))}$ bounded adversary.) Then, there is an efficiently computable boolean function f_R with the following properties. If Alice and Bob are computationally unbounded, then there is an $O(\log n)$-bit 1-round randomized protocol that computes f_R. But if Alice and Bob are computationally bounded, then any randomized protocol for f_R, even with multiple rounds, will require $\Omega(T(n))$ bits of communication.

As a corollary we get the following strong separation result. Let F_c denote the class of functions $f(x, y) \in$ PTIME such that the randomized communication complexity of f is bounded by c. Similarly, let F_c^{poly} be the functions $f(x, y) \in$ PTIME such that $f(x, y)$ is computable by polynomial-time parties with communication c. Then there is an explicit boolean function f in $F_{\log n} \setminus F_{T(n)}^{\text{poly}}$ for $T(n)$ as above.

Deterministic communication model. Obtaining similar tradeoff results for the deterministic two-party model appears to be much harder. We show a strong tradeoff result relative to a random oracle. Specifically, let L be a random sparse language. Then, with probability 1 over choice of L, there is a boolean function f_L (efficiently computable relative to L) with the following properties. There is a *deterministic* communication protocol for f_L with, say, $O(\log^2 n)$ bits of communication if both Alice and Bob are computationally unbounded with oracle access to L. However, any protocol in which Alice and Bob are computationally bounded will require $\Omega(n)$ bits of communication, even with oracle access to L.

Query complexity and property testing. Our next results prove tradeoffs in related models like the query complexity model and the property testing model. In these models, information is stored in the form of a table and the queries are answered by bit-probes to this table. We view the probes as communication between the stored table and the query scheme (or the tester), and the computation of the query scheme (or the tester) as the local computation. We show that: (a) Under a cryptographic assumption, there exists a language L such that, on inputs of length n, a query scheme with unlimited computation makes $O(\log n)$ queries while a query scheme with efficient local computation requires $\Omega(n^\varepsilon)$ queries for some fixed $\varepsilon < 1$; (b) assuming NP$\not\subseteq$ BPP, given any $\varepsilon > 0$, there exists a property P such that, on inputs of length n, a computationally unbounded tester will require only n^ε bits to check if the input satisfies the property or is far from satisfying it. On the other hand, a computationally bounded tester will require $n^{1-\varepsilon}$ bits. This result can be strengthened to any sub-exponential tradeoff under a stronger assumption that all languages in NP do not have randomized sub-exponential time algorithms.

Natural tradeoff questions. In addition to proving the *existence* of tradeoffs in various contexts, we also put forward several concrete *natural* tradeoff questions and relate them to each other. We propose three different tradeoff questions arising in different contexts: arithmetization of boolean functions, multi-party communication, and cryptography. We relate them by showing that a "positive" resolution of the first would imply a solution to the second, which in turn would imply a solution to the third. Hence, the cryptographic application may serve as an additional motivation for studying the other two.

2 Preliminaries

In this section, we describe the communication complexity model, a formal definition of the problem we consider and the notion of UP relations.

2.1 The Communication Complexity Model [Yao86]

Let X, Y and Z be arbitrary finite sets and $f : X \times Y \to Z$ be an arbitrary function. There are two players, Alice and Bob who wish to evaluate $f(x, y)$ for $x \in X$ and $y \in Y$. However, Alice only knows x and Bob only knows y. To evaluate the function, they communicate with each other according to some fixed protocol P in which they send messages to each other.

The cost of a protocol P on an input (x, y) is the number of bits exchanged by Alice and Bob when Alice is given x and Bob is given y. The cost of a protocol P is the worst case cost of P over all inputs (x, y). The (deterministic) communication complexity of f is the minimum cost of a protocol that computes f.

If Alice and Bob are allowed access to random coin tosses and their messages depend also on the result of the coin tosses besides their input and the communication so far, we say that the protocol P is randomized. The randomized communication complexity of a function f is the minimum cost of a randomized protocol that computes f with error at most $\frac{1}{4}$ on any input (x, y). The error is over the internal coin tosses of the protocol.

2.2 Tradeoffs

We now describe formally our tradeoff problem in the two-party communication complexity model. Similar definitions can be given for other models we consider. Our goal is to find a boolean function $f : X \times Y \to \{0, 1\}$ with the following properties:

- $f(x, y)$ can be computed efficiently, that is in polynomial time, if both the inputs $x \in X$ and $y \in Y$ are given.
- f has very efficient communication protocols, that is, protocols with communication complexity $(\log n)^c$ for some c.
- There is no protocol for f which is simultaneously communication and computation efficient. In other words, any protocol in which Alice and Bob use only polynomial time for local computation requires almost linear number of bits of communication in the worst case.

2.3 UP Relations

Definition 2.1 *A relation $R \subseteq \Sigma^* \times \Sigma^*$ is said to be a* UP relation *(with witness size n^k) if*

1. *there exists a deterministic Turing machine that decides the language $\{(x, w) | (x, w) \in R\}$ in polynomial time.*
2. *for every x, there exists at most one w such that $(x, w) \in R$ and furthermore, this w satisfies $|w| = |x|^k$. We denote this w, if it exists, by $w(x)$.*

The search problem *corresponding to R is the problem of finding w such that $R(x, w)$ holds, given x.*

We will assume the existence of UP relations for which the corresponding search problem is very hard. Such an assumption is standard in cryptography since the existence of strong one-way permutations implies the existence of such hard UP relations. More formally,

Definition 2.2 *We say is that a UP relation R is $T(n)$-hard if no probabilistic algorithm running in time $2^{O(T(n))}$ solves the search problem corresponding to R.*

3 Tradeoffs in the Two-Party Communication Complexity Model

3.1 Randomized Model

We start with the definition of the boolean function we consider.

Definition 3.1 *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a UP relation with witness size n^k. Consider the 2-player (Alice and Bob) boolean function $f_R : \{0,1\}^{n+n^k} \times \{0,1\}^{n^k} \to \{0,1\}$.*

$$
\begin{aligned}
\textit{Alice's input:} \quad & (x,z) \in \{0,1\}^n \times \{0,1\}^{n^k} \\
\textit{Bob's input:} \quad & w \in \{0,1\}^{n^k} \\
f_R((x,z),w) \quad = \quad & \begin{cases} \langle z, w\rangle & \textit{if } R(x,w) \textit{ holds} \\ 0 & \textit{otherwise} \end{cases}
\end{aligned}
$$

where $\langle a, b\rangle$ denotes the inner product of a, b modulo 2.

Theorem 3.2 *Let R be a $T(n)$-hard UP reaction. Then, the predicate f_R has the following properties.*

1. *f_R is computable in polynomial time.*
2. *There exists a randomized protocol that computes f_R with $O(\log n)$-bit communication.*
3. *If Alice and Bob are computationally bounded, then any randomized protocol for f_R, even with multiple rounds, will require $\Omega(T(n))$ bits of communication.*

Proof: Observe that f_R can be computed efficiently given both its inputs. We just need to check that $R(x,w)$ holds and if so, output $\langle z, w\rangle$.

Lemma 3.3 *If Alice is computationally unbounded, then there exists a randomized protocol that computes f_R with $O(\log n)$-bit communication.*

Proof: Alice computes the unique w such that $R(x,w)$ holds. Alice and Bob then engage in an "equality" protocol[3] to check that Bob's input equals w. If so, she computes and sends Bob the answer $\langle z, w\rangle$. □ □

The following lemma demonstrates that such a communication-efficient protocol is unlikely when Alice and Bob are computationally bounded. In fact, it is sufficient for the proof that only Alice is computationally bounded. Bob is allowed to be computationally unbounded.

[3]Recall that the randomized communication complexity of equality is $O(\log n)$.

Lemma 3.4 *Suppose there exists a $b(n)$-bit communication randomized multi-round protocol Π that computes f_R involving Alice whose running time is at most $T_A(n)$, then there exists a randomized algorithm that solves the search problem corresponding to R in time $\mathrm{poly}(n, 2^{b(n)}) \cdot T_A(n)$.*

Proof:

For the rest of the argument, we assume that for any x, w is the unique w such that $R(x, w)$ holds, denoted by $w(x)$. Hence, for our purposes, $f_R((x, z), w) = \langle z, w \rangle$.

Our goal is to relate the search problem of computing w given x to the problem of computing $\langle z, w \rangle$ with a low communication protocol. Our approach is to convert a low communication protocol into an efficient oracle that computes $\langle z, w \rangle$ with some advantage over random guessing. Given such an oracle, we can then use the Goldreich-Levin reconstruction algorithm to compute a small number of candidates for w. More precisely, we create a "small" set of oracles, one of the oracles computes $\langle z, w \rangle$ with some nontrivial advantage. We try each oracle by exhaustive search, and use the fact that we can recognize the correct w.

Converting protocols into oracles

Let $\mathcal{T}$ be a transcript. For simplicity, we assume Alice outputs $f_R((x, z), w)$ as its final bit; this convention increases the size of the transcript by at most 2 bits. Thus, $\mathcal{T}$ includes a "guess" as to $\langle z, w \rangle$. We define the probabilistic oracle $A_{\mathcal{T}}(x, z)$ for computing $\langle z, w \rangle$, as follows.

Algorithm $A_{\mathcal{T}}$ (Input: $(x, z) \in \{0,1\}^n \times \{0,1\}^{n^k}$).

> Simulate the protocol Π from Alice's end. Whenever a message from Bob is required, use the transcript $\mathcal{T}$ to obtain the corresponding message. If at any point the message generated by Alice according to the protocol Π disagrees with the contents of the transcript $\mathcal{T}$, abandon the protocol and output a random bit b. Otherwise, follow the protocol to the end and output the bit b generated by the protocol Π.

First we define our notation for the advantage of Π and $A_{\mathcal{T}}$ in guessing $\langle z, w \rangle$.

Definition 3.5 *Let $x \in \{0,1\}^n$, $w = w(x)$ and z be distributed uniformly. We define $\textsc{adv}(\Pi, x)$ by*

$$\textsc{adv}(\Pi, x) = \Pr[\textit{Alice outputs } \langle z, w \rangle] - \Pr[\textit{Alice doesn't output } \langle z, w \rangle],$$

where Alice and Bob run Π with respective inputs (x, z) and w, and the probability is taken over the choice of z and over the coin tosses of Alice and Bob. We define $\textsc{adv}(A_{\mathcal{T}}, x)$ analogously. Fixing x and a transcript $\mathcal{T}$, we define $\textsc{adv}(\Pi, x, \mathcal{T})$ by

$$\begin{aligned} \textsc{adv}(\Pi, x, \mathcal{T}) = \ & \Pr[\mathcal{T} \textit{ occurs and Alice outputs } \langle z, w \rangle] \\ & - \Pr[\mathcal{T} \textit{ occurs and Alice doesn't output } \langle z, w \rangle] \end{aligned}.$$

Note that the only contribution to $A_{\mathcal{T}}$'s advantage is by events in which $\mathcal{T}$ occurs, hence we do not bother to define $\textsc{adv}(A_{\mathcal{T}}, x, \mathcal{T})$. It follows from the definitions that,

$$\textsc{adv}(\Pi, x) = \sum_{\mathcal{T}} \textsc{adv}(\Pi, x, \mathcal{T}). \tag{1}$$

Since the protocol Π computes f_R correctly, it holds that $\textsc{adv}(\Pi, x) \geq \frac{1}{2}$ for every x. Since there are at most $2^{2b(n)}$ possible transcripts $\mathcal{T}$, it follows from Equation (1) that for every $x \in \{0,1\}^n$, there exists a transcript $\mathcal{T}^*$,

$$\textsc{adv}(\Pi, x, \mathcal{T}^*) \geq \frac{1}{2^{2b(n)+1}} \tag{2}$$

A simple but key observation is that we can often bound $\textsc{adv}(A_{\mathcal{T}}, x)$ by $\textsc{adv}(\Pi, x, \mathcal{T})$.

Proposition 3.6 *If* $\text{ADV}(\Pi, x, \mathcal{T}) > 0$ *then* $\text{ADV}(A_{\mathcal{T}}, x) \geq \text{ADV}(\Pi, x, \mathcal{T})$.

Proof: Let $\rho_w^{(x,z)}(\mathcal{T})$ be the probability that Alice and Bob's coins are consistent with $\mathcal{T}$ when running Π with respective inputs (x, z) and w. Likewise, let $\rho^{(x,z)}(\mathcal{T})$ be the probability that Alice's coins are consistent with $\mathcal{T}$. Thus, $\rho^{(x,z)}(\mathcal{T})$ is the probability that $\mathcal{T}$ occurs when running $A_{\mathcal{T}}(x, z)$. Finally, let $\rho_w(\mathcal{T})$ be the probability that Bob's coins are consistent with $\mathcal{T}$. Note that $\rho_w(\mathcal{T})$ is independent of z. It follows from the definitions that

$$\rho_w^{(x,z)}(\mathcal{T}) = \rho^{(x,z)}(\mathcal{T})\rho_w(\mathcal{T}). \tag{3}$$

Fixing x (and $w = w(x)$), let G denote the set of z such that $\mathcal{T}$ gives the correct value for $\langle z, w\rangle$, and B denote the set of z such that gives the incorrect value. Then we can express $\text{ADV}(\Pi, x, \mathcal{T})$ and $\text{ADV}(A_{\mathcal{T}}, x)$ by

$$\text{ADV}(\Pi, x, \mathcal{T}) = \frac{1}{2^{n^k}} \left(\sum_{z \in G} \rho_w^{(x,z)}(\mathcal{T}) - \sum_{z \in B} \rho_w^{(x,z)}(\mathcal{T}) \right) \text{ and} \tag{4}$$

$$\text{ADV}(A_{\mathcal{T}}, x) = \frac{1}{2^{n^k}} \left(\sum_{z \in G} \rho^{(x,z)}(\mathcal{T}) - \sum_{z \in B} \rho^{(x,z)}(\mathcal{T}) \right). \tag{5}$$

From Equations (3), (4) and (5), and $\rho_w^{(x,z)}(\mathcal{T}) = \rho^{(x,z)}(\mathcal{T})\rho_w(\mathcal{T})$, it follows that

$$\text{ADV}(\Pi, x, \mathcal{T}) = \text{ADV}(A_{\mathcal{T}}, x)\rho_w(\mathcal{T}). \tag{6}$$

Since $0 \leq \rho_w(\mathcal{T}) \leq 1$, the proposition follows (note that when $\rho_w(\mathcal{T}) = 0$, $\text{ADV}(\Pi, x, \mathcal{T}) = 0$). □ □

Fix any $x \in \{0, 1\}^n$. Consider the transcript $\mathcal{T}^*$ guaranteed to exist by Equation (2). For this transcript, we conclude from Proposition 3.6 that

$$\text{ADV}(A_{\mathcal{T}^*}, x) \geq \frac{1}{2^{2b(n)+1}}. \tag{7}$$

Set $\varepsilon = \frac{1}{2^{2b(n)+1}}$. Now we run the Goldreich-Levin algorithm GL (See Theorem 3.7) with parameters n, ε, oracle access to $A_{\mathcal{T}^*}(x, .)$ and predicate $R(x, .)$.

Theorem 3.7 (Goldreich-Levin [GL89]) *There exists a randomized algorithm GL with oracle access to a function and a predicate satisfying the following: Fix* $u \in \{0, 1\}^n$. *Let* $h : \{0, 1\}^n \to \{0, 1\}$ *be a randomized algorithm such that* $h(v) = \langle u, v\rangle$ *with probability at least* $\frac{1}{2} + \varepsilon$ *where the probability is over choice of* v*, picked uniformly at random, and the internal coin tosses of* h*. Let* $P : \{0, 1\}^n \to \{0, 1\}$ *be a polynomial time computable predicate such that* $P(v) = 1$ *iff* $u = v$*. Then, the randomized algorithm GL with oracle access to* h *and* P *satisfies*

$$\Pr[GL^{h,P}(n, \varepsilon) = u] \geq \frac{3}{4}$$

Moreover, the running time of GL is at most $\text{poly}(n, \frac{1}{\varepsilon})$.

Theorem 3.7 guarantees that the algorithm GL computes w in time $\text{poly}(n, 1/\varepsilon)$ with constant probability. However, we do not have the transcript $\mathcal{T}^*$. (Recall that we only know that there exists a transcript $\mathcal{T}^*$ that satisfies Equation (7), we do not how to obtain one.) For this purpose, we run the Goldreich-Levin algorithm GL for every possible transcript $\mathcal{T}$ with parameters n and ε. One of these must succeed. Moreover, we can check which one succeeds by verifying that $R(x, w)$ holds. The total time taken by this algorithm is at most $2^{2b} \cdot \text{poly}(n, 2^{2b+1}) \cdot T_A(n) = \text{poly}(n, 2^b) \cdot T_A(n)$. This proves Lemma 3.4. □ □ To conclude the proof of the tradeoff result, we now use the assumption that the search problem corresponding to UP relation R does not have randomized algorithm that run in time $2^{o(T(n))}$ on inputs of length n. Therefore, $\text{poly}(n, 2^b) \cdot T_A(n) \geq 2^{\Omega(T(n))}$ and hence $b(n) = \Omega(T(n))$ since $T_A(n)$ is polynomially bounded in n. □ □

Remarks:

1. If we make the assumption that there is a search problem in UP that does not have sub-exponential time randomized algorithms, we get a very strong tradeoff. Such an assumption is used in cryptography.

2. We can prove the same result under a weaker assumption that the class FewP has a relation whose search problem is hard. In this case, we could use the set membership function instead of equality.

3. If the search problem corresponding to the relation R had average-case complexity at least $2^{\Omega(T(n))}$ when x is chosen from the distribution $\mathcal{D}$ (instead of worst case complexity), then the same proof as above demonstrates that f_R has average-case communication complexity at least $\Omega(T(n))$ for polynomially bounded Alice and Bob when x is chosen from the distribution $\mathcal{D}$, z uniformly and $w = w(x)$.

3.2 Deterministic Model

Curiously, it seems harder to obtain strong tradeoff results for the deterministic model. This is due, in part, to to the weakness of this model (e.g. one cannot implement an efficient equality checking protocol). For this model, we show tradeoff results relative to a random oracle. We stress that we do not use the random oracle to 'randomize' the protocol. In particular, we require our protocols not to produce errors.

We first give a definition of the Boolean function that we consider.

Definition 3.8 *Let L be any language. Consider the 2-player (Alice and Bob) boolean function $f_L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.*

$$\begin{aligned} \textit{Alice's input:} &\quad x \in \{0,1\}^n \\ \textit{Bob's input:} &\quad y \in \{0,1\}^n \\ f_L(x,y) &= \begin{cases} \langle x, y \rangle & \textit{if } x \textit{ and } y \textit{ are in } L \\ 0 & \textit{otherwise} \end{cases} \end{aligned}$$

where $\langle x, y \rangle$ denotes the inner product of x, y modulo 2.

Theorem 3.9 *Let L be a random sparse language that contains $2^{k(n)}$ strings of length n. Here, $k(n)$ can be any function such that $k(n) = \omega(\log n)$. Then, with probability 1 over choice of L, the predicate f_L has the following properties given an oracle access to L:*

1. *f_L can be computed in polynomial time.*

2. *f_L has a communication protocol with $k(n)+1$ bits of communication.*

3. *Any communication protocol (deterministic or randomized) in which Alice and Bob are computationally bounded requires $\Omega(n)$ bits of communication (even with oracle access to L). Note that the choice of protocol may depend on the choice of L.*

Property 1 is easy to verify: to compute $f_L(x,y)$ it suffices to check if x and y belong to L using the oracle access, and then compute their inner product if necessary. Next we prove Property 2.

Lemma 3.10 *If Alice and Bob are computationally unbounded and have oracle access to L, then f_L has a communication protocol with $k(n)+1$ bits of communication.*

Proof: Alice, using oracle access to L, determines if x belongs to L or not. If it does, she also finds out the position of x in the lexicographic ordering of strings in L. She sends this information to Bob who outputs $\langle x, y\rangle$ if both x and y belong to L and 0 otherwise. $\Box$ $\Box$ In the remainder of this section we establish Property 3.

Lemma 3.11 *With probability* 1 *over the choice of L, any communication protocol for f_L in which Alice and Bob run in time $2^{o(k(n))}$ will require $\Omega(n)$ bits of communication, even with oracle access to L and even if the protocol is allowed to be randomized and err with a small probability.*

The proof of Lemma 3.11 proceeds as follows. We start with a protocol which is efficient with respect to both communication and computation. We first show that oracle calls are not of much use for such protocols. In other words, we can get another protocol with similar efficiency which does not make oracle calls to L. The intuition for this step is that a time-bounded protocol is unlikely to query L on any input $z \in L$ which is different from its inputs (x, y). Next we use the fact that the inner product function is hard on the average for low-communication protocols. Hence, the protocol we obtain defines a procedure that distinguishes a pair of random elements from L from a pair of truly random strings *without access to L*. Since L is a random sparse language, this leads to a contradiction.

The details of the proof can be found in Appendix A.

4 Tradeoffs in Related Models

4.1 Communication vs. Computation in the Query Complexity Model

We consider the query complexity model in which a decision procedure D probes its input x choosing to look at some bits, but not others. The query complexity of a predicate P on n-bit inputs is given by

$$\min_D \max_x (\#\text{ probes } D \text{ makes on } x)\,.$$

Here, D ranges over all decision procedures for P and x ranges over all inputs of length n.

We can consider the computationally bounded analog of this measure, where D is restricted to run in probabilistic polynomial time. Some subtleties arise in such a definition. For example, D must be quantified before n, since polynomial time is an asymptotic notion, but under this quantification there may be no "best" D for all inputs. Also, we may wish to augment our definitions to allow for an error probability.

Fortunately, Theorem 4.2 establishes a tradeoff that is clearly resilient to these technical issues.

Definition 4.1 *We define $lsb_\ell(x)$ to be ℓ least significant bits of x. We say that a one-way permutation p is $\ell(n)$-lsb hard if no probabilistic polynomial-time procedure, on input x, can compute $lsb_{l(n)}(p^{-1}(x))$ with probability non-negligibly greater than $2^{-\ell(n)}$, where x is chosen uniformly from $\{0,1\}^n$.*

We note that such permutations exist based on the hardness of computing discrete logarithms over composite integers [SS90, HSS93].

Theorem 4.2 *Let p be $\ell(n)$-lsb hard. Then there exists a predicate*

$$C_p : (\{0,1\}^n)^{2^{\ell(n)}+1} \longrightarrow \{0,1\}$$

with the following properties:

1. *C_p is computable in polynomial time.*
2. *The query complexity of C_p is at most $2n$.*
3. *No polynomial-time bounded decision procedure Q can compute C_p querying only $2^{\alpha\ell(n)}$ bits, where $\alpha < 1$ is any constant. In particular, there is a distribution on the inputs so that if Q computes C_p with advantage ε, then one can compute $lsb_{\ell(n)}(x)$ from $p(x)$ with probability $\Omega(\varepsilon 2^{-\alpha\ell(n)})$.*

Proof: (Sketch) For notational simplicity, we write ℓ instead of $\ell(n)$. We define $C_p(y, x_1, \ldots, x_{2^\ell})$ to be 1 iff there exists some i, $1 \leq i \leq 2^\ell$, such that $p(x_i) = y$ and $lsb_\ell(x_i) = i$ (treating i as an ℓ-bit string).

The predicate C_p is computable in polynomial time, since we can run over all the (polynomially-many) possible values of i. To see that C_p has query complexity at most $2n$, consider the following (computationally unbounded decision procedure):

1. Query y (which is n bits long)
2. Compute $x = p^{-1}(y)$ and $i = lsb_\ell(x)$.
3. Query x_i (which is n bits long), and accept iff $x_i = x$.

Our proof that no polynomial-time bounded decision procedure exists is by contradiction. Given Q, as above, we construct a polynomial-time algorithm G for guessing $lsb_\ell(x)$ from $p(x)$, as follows:

1. Given $p(x)$, compute $y = p(x)$ and choose $x_1, \ldots, x_{2^\ell}$ uniformly at random from $\{0,1\}^n$.
2. Run Q on input $(y, x_1, \ldots, x_{2^\ell})$. Define I by

$$I = \{i : Q \text{ queries at least one bit of } x_i\}.$$

3. Choose a random index i from I and output i (as an ℓ-bit quantity).

We relate the success probability of G to Q's advantage, ε at computing $C_p(y, x_1, \ldots, x_{2^\ell})$ under the distribution of inputs obtained as follows:

1. Choose x uniformly from $\{0,1\}^n$, and let $y = p(x)$ and $i = lsb_\ell(x)$.

2. For $j \neq i$, choose x_j uniformly from $\{0,1\}^n$.

3. With probability 1/2, choose $x_i = x$ (the predicate is true). Else, choose x_i uniformly from $\{0,1\}^n - x$ (the predicate is false).

Clearly, if on a particular run, Q never queries any bit in x_i, it has no advantage in guessing the value of the predicate. It follows that with probability $\Omega(\varepsilon)$, $i \in I$, where I is defined as above. In this case, choosing from I uniformly will yield i with probability $1/|I|$. Since $I \leq 2^{\alpha\ell(n)}$, the theorem follows. □ □

Our construction only assumes that $p()$ is strong against polynomial adversaries, resulting in any polynomial tradeoff. With stronger assumptions on the simultaneous hardness of bits in $p()$, we can prove any sub-exponential tradeoff.

4.2 Communication vs. Computation for Property Testing

The property testing model is very similar to the query complexity model defined above. Given an input x, the tester probes certain bits of x and we count the number of such queries. Let P be a predicate on n-bit inputs. A ε-tester for P is a randomized procedure that should output 1 on inputs x satisfying P with probability at least 3/4 and should output 0, with probability 3/4, on inputs x that are ε-far from any input x' satisfying P. The output of the tester on other instances of length n is arbitrary. The ε-testing complexity of a predicate P on inputs of length n is given by

$$\min_T \max_x (\# \text{ probes } T \text{ makes on } x)\,.$$

Here, T ranges over all possible testers for P and x ranges over all inputs of length n.

Theorem 4.3 *Let L be any language such that $L \in NP$ and $L \notin BPP$. Fix any $\varepsilon > 0$. Fix any $c > 1$ and let $m = n^c$. Then, there exists a predicate*

$$P_L : (\{0,1\}^n)^{2m} \longrightarrow \{0,1\}$$

with the following properties:

1. *P_L is computable in polynomial time.*

2. *The ε-testing complexity of P_L is $O(n)$.*

3. *No polynomial-time bounded property ε-tester T can compute P_L querying less than m bits.*

Proof: To define the predicate P_L, we need to introduce some notation. We start with a language L in NP that is not in BPP and let $R_L(x,w)$ its corresponding relation. Without loss of generality, let us assume that $|x| = |w|$ and denote it n. Let $C : \{0,1\}^n \to \{0,1\}^{mn}$ be an error correcting code with distance greater than $2\varepsilon mn$. For $y \in \{0,1\}^{mn}$ and $w_1, \ldots, w_m \in \{0,1\}^n$, we define our property $P_L(y, w_1, \ldots, w_m)$ to be 1 if $y = C(x)$ for some $x \in L$, and $R_L(x, w_1 \oplus \cdots \oplus w_m)$ holds (i.e. if $w_1, \ldots, w_m$ compose to give a witness for $x \in L$).

The predicate P_L is computable in polynomial time since we can decode y to get x and then check that $w_1, \ldots, w_m$ do compose to give a witness for the membership of x in L. To check that the ε-testing complexity of P_L is at most n, consider the following (computationally unbounded) tester:

1. Probe $O(n)$ random locations of y.

2. Check if y agrees with $C(x)$ for some x. If not, reject.

3. Otherwise, check if x is in L and if so, accept.

It is easy to check that the tester outputs 1 with high probability on all instances for which P_L holds. To show that the output of the tester is correct on inputs that are ε-far from satisfying P_L, we make the following simple observation. If y encodes $x \in L$ then it is possible to change only w_1 (i.e. $n \ll \varepsilon mn$ bits) to get an instance that satisfies P_L. Therefore, instances that are ε-far from an instance that satisfies P_L must either contain y that ε-far from any $y = C(x)$ for $x \in L$. Hence, our computationally unbounded tester decides the property without looking at $w_1, \ldots, w_m$. The tester probes n bits of y and rejects if they do not agree with any codeword $y = C(x)$ for $x \in L$. The probability of giving a wrong answer is that of two codewords agreeing on $O(n)$ random locations.

Suppose that a polynomial time bounded tester T exists that queries less than m bits. We will use T to design a probabilistic polynomial time algorithm D for membership in L.

1. Given an input x, compute $y = C(x)$.

2. Choose m random strings $w_1, \ldots, w_m$ in $\{0,1\}^n$.

3. Output $T(y, w_1, \ldots, w_m)$.

Since T reads less than m bits of the input, we note that it has no information on even a single bit of the possible witness $w = w_1 \oplus \cdots \oplus w_m$. Therefore, the algorithm D has the same probability of success as T. This implies that L is in BPP, a contradiction. □ □

Our construction only uses the fact that NP $\not\subseteq$ BPP. A stronger assumption that all languages in NP do not have sub-exponential time randomized algorithms enables us to obtain any sub-exponential tradeoff.

5 Natural Tradeoff Questions

In contrast to previous sections which focus on proving the *existence* of tradeoffs, in this section we put forward several concrete *natural* tradeoff questions and relate them to each other.

We propose three different tradeoff questions arising in different contexts: arithmetization of boolean functions, multiparty communication, and cryptography. We relate them by showing that a (positive) solution to the first would imply a solution to the second, which in turn would imply a solution to the third.

5.1 Time vs. Degree in Arithmetization

Arithmetization of boolean functions has proved to be a very powerful tool in complexity theory [LFKN92, BF91]. A multivariate polynomial $p(X_1, \ldots, X_n)$ over a field[4] F is said to *extend* a boolean function $f : \{0,1\}^n \to \{0,1\}$ if p and f agree on $\{0,1\}^n$; that is, if for any $x \in \{0,1\}^n$ we have $p(x) = f(x)$. Any function f admits a unique *multi-linear* extension, i.e., an extension

[4] In what follows fields can be replaced by rings, and polynomials over fields by polynomials over the integers. We prefer to use polynomials over (finite) fields because of their use in applications.

in which the degree in each variable is at most 1. If the degree bound is relaxed, the number of extensions becomes large.

We consider the computational complexity of evaluating a polynomial extension at a given point in F^n. Specifically, given a function f and a bound d on the degree in each variable, how easy is the computationally easiest extension meeting the degree constraint? In the case $d = 1$, it is easy to see that the (multi-linear) extension of any function f is in EXPTIME^f. But if the function f is "easy" (say, in AC^0) can anything better be said about the time complexity of its multi-linear extension? Also, if we relax the degree bound to $d > 1$, does this allow for more time-efficient extensions? As we shall see, it is not hard to answer these questions in a way that establishes some weak tradeoff between time and degree.

Towards a more concrete study of the time-degree tradeoff question we formulate the following nonuniform algebraic version of the problem.

Definition 5.1 *Given a boolean function $f : \{0,1\}^n \to \{0,1\}$ and a bound d on the degree (in each variable), define $\mathsf{TD}(f, d)$ to be the minimal integer s such that for any field F there is a size-s algebraic circuit over F evaluating some degree-d extension of f. We will also allow the first argument to TD to be a function $f : \{0,1\}^* \to \{0,1\}$, in which case the degree bound will be a function of the input length n and the output will also be a function of n.*

We first argue that if the degree bound d is as large as the formula size of f, then the time complexity of the extension is bounded by formula size.

Claim 5.2 *Suppose $f : \{0,1\}^n \to \{0,1\}$ can be computed by a boolean formula of size ℓ. Then $\mathsf{TD}(f, \ell) = O(\ell)$.*

Proof: A degree-ℓ extension as required is obtained by the straightforward arithmetization of a formula φ of size ℓ computing f. For instance, if $\varphi = \varphi_1 \vee \varphi_2$ then the polynomial p_φ is recursively defined by $p_\varphi = 1 - (1 - p_{\varphi_1})(1 - p_{\varphi_2})$. In fact, even the *total* degree of this extension is bounded by ℓ. □ □

Note that if a formula is replaced by a circuit, then the degree of the above straightforward arithmetization may become exponential in the circuit size.

We now argue that if one insists on a minimal degree extension, then even for some very easy functions f evaluating their extension may become $\#P$-hard. This gives a provable, albeit very weak, tradeoff between time and degree under a standard complexity assumption.

Claim 5.3 *There is an AC^0 function f such that if $\mathsf{TD}(f, 1) = n^{O(1)}$ then $P = \#P$.*

Proof sketch: Let f be a universal function for 3CNF formulas. That is, $f(G; x) = G(x)$, where the G variables represent a 3CNF formula on n inputs. The function f can be computed in AC^0. Let F be a field of characteristic different than 2,[5] and let $\beta = 2^{-1}$ (so that $\beta = 1 - \beta$). Let p be the (unique) multi-linear extension of f over a field F. Using an oracle to p, one can easily count the number of satisfying assignments of a given formula g modulo the characteristic of F. This is done by evaluating $p(g; \beta, \beta, \ldots, \beta)$ and dividing the result by β^n. Note that each nonzero term in the multi-linear extension corresponds to some satisfying assignment of g. Since $\beta = 1 - \beta$, each such term evaluates to β^n. □

[5]This restriction is not necessary, but it makes the proof a little simpler.

There are various questions one could ask regarding time-degree tradeoffs. In particular, one could try to improve the degree bounds $\ell, 1$ in Claims 5.2, 5.3 respectively, ideally closing the gap between the two. For concreteness, we would like to highlight the following special case of the general question.

Question 5.4 *Does every poly-time computable f admit a polynomial degree bound $d(n)$ such that $\mathsf{TD}(f,d) = n^{O(1)}$? Can $d(n)$ as above be made independent of f?*

5.2 Time vs. Communication in Multi-party Simultaneous Messages Protocols

We begin by presenting a solution to the riddle from the Introduction. We then extend the riddle to a more general problem which might exhibit a time-communication tradeoff, and relate it to the time-degree problem described above.

Solving the riddle. Let s_i denote the sum of the entries in the ith row of M. We show how $k = n+1$ players can communicate $\mathsf{PS}(M) = \prod_{i=1}^{n} s_i$ to the referee by each sending a single, *efficiently computable* element of F. (The same solution will work for any larger number of players.) The high-level idea is to first convert the "additive" representation of s_i to a degree-1 polynomial representation over a sufficiently large extension field, then make each player locally multiply its values of the n polynomials (one for each s_i), and finally project down to the original field. The protocol's outline is described below.

1. Each entry of M is lifted to an extension field F' of F such that $|F'| \geq k+1$. (This is only a conceptual step and requires no action, since F is a subfield of F'.) Let $\alpha_1, \ldots, \alpha_k$ be distinct nonzero elements of F'.

2. The players locally process their entries of M, and each outputs a single element of F' for each row. Let $P_{i,j}$ denote the output of player j corresponding to the ith row. The values $P_{i,j}$ should satisfy the following requirement: for each i, the k points $(\alpha_j, P_{i,j})$ lie on a degree-1 polynomial over F' whose free coefficient is s_i. The implementation of this stage will be described below.

3. Each player j multiplies its n local outputs $P_{i,j}$ from the previous state, resulting in a single element $q_j \in F'$. Note that the k points (α_j, q_j) now lie on a degree-n polynomial whose free coefficient is precisely $\prod_{i=1}^{n} s_i = \mathsf{PS}(M)$. Since $k > n$, this polynomial can be uniquely determined by interpolation and its free coefficient can be written as $\sum_{j=1}^{k} \lambda_j q_j$ for some fixed coefficients $\lambda_j \in F'$. Each player j projects $\lambda_j q_j$ down to the original field using a field homomorphism $h : F' \rightarrow F$, and sends the result to the referee.

4. The referee outputs the sum of the k field elements it received.

It remains to describe the implementation of Step 2. Define a $k \times k$ matrix L over F' such that $L_{\ell,m} = 1 - \frac{\alpha_\ell}{\alpha_m}$. For each i, we let $P_{i,j} = \sum_{m=1}^{k} L_{j,m} M_{i,m}$. Note that since $L_{j,j} = 0$, player j can compute this sum based on his local input. It remains to argue that the above local computations indeed produce the required degree-1 representation of s_i. This follows by noting that for any column m of L, the values $(\alpha_\ell, L_{\ell,m})$ lie on a degree-1 polynomial whose free coefficient is 1. By linearity, the values $(\alpha_j, P_{i,j})$ lie on a degree-1 polynomial whose free coefficient is $\sum_{j=1}^{k} 1 \cdot M_{i,j} = s_i$. Thus, we have shown:

Theorem 5.5 *The function* $\mathsf{PS}(M) = \prod_{i=1}^{n} \sum_{j=1}^{k} M_{ij}$, *where* $k > n$, *admits a* computationally efficient *simultaneous messages protocol in which each player holds all but one column of* M *and sends a single field element to the referee.*

Tradeoff candidates. Given the easiness of the product-of-sums question, and motivated by the application in Section 5.3, we would like to consider the following generalization. Instead of computing the product of the row sums s_i, we now allow an arbitrary polynomial-time computable function $f(s_1, \ldots, s_n)$. (For simplicity, assume that f is a boolean function and we are promised that $s_i \in \{0, 1\}$.) Note that without any restriction on the time complexity, one can obtain a communication efficient protocol by combining the previous protocol with the multi-linear extension of f. However, this protocol is not computationally efficient. What we view as the most interesting instance of the general question is the following.

Question 5.6 *Can the generalized problem be solved for any poly-time* f *using* $k = \mathrm{poly}(n)$ *players, efficient communication (a single field element per player), and poly-time computation? Can this be achieved with* k *being independent of the (polynomial) time complexity of* f*?*

The following is obtained by a straightforward generalization of the solution to the riddle presented above.

Claim 5.7 *A positive answer to Question 5.4 implies a positive answer to Question 5.6.*

5.3 Time vs. Communication and Rounds in Cryptography

In the problem of private computation [Yao86, GMW87, BGW88, CCD88], there are k players who wish to jointly compute some function f on their inputs without revealing their inputs to each other. The players are allowed to communicate over secure channels according to some prescribed protocol. By default, we require that at the end of the protocol all players learn the value of f, but no individual player can learn any additional information from the messages it received (other than what follows from its own input and the value of f).

Standard protocols for this problem (e.g., [BGW88, CCD88]) allow to compute an arbitrary function f. Given a circuit representation for f, their time complexity is proportional to the circuit size and their round complexity to its depth. It is also known [BFKR90] that an *arbitrary* function $f : \{0, 1\}^n \to \{0, 1\}$ can be privately computed by $k = n/\log n$ players[6] (or more) in a constant number of rounds using polynomial communication. Moreover, in this protocol the (polynomial) amount of communication does not depend on the hardness of f. However, due to its use of a multi-linear extension of f, the protocol is not computationally efficient even if f is. Thus, we have a natural cryptographic candidate for tradeoffs between time and communication and rounds.

Question 5.8 *Can any poly-time computable* f *be privately computed by* $\mathrm{poly}(n)$ *players using polynomial computation and a constant number of rounds? Can the communication complexity be made independent of the (polynomial) time complexity of* f*?*

While it is possible to directly derive a positive answer to the above question from a positive answer to Question 5.4, one can in fact make the following stronger claim.

[6] Here and in the following, the n input bits may be arbitrarily partitioned between the players.

Claim 5.9 *Any solution to Question 5.6 in which the referee outputs the sum of the k field elements it receives implies a positive answer to Question 5.8.*

Proof sketch: Let k be the (polynomial) number of players guaranteed by the solution to Question 5.6. Assume (without loss of generality) that there are 3 disjoint sets of players. Players P_i, $1 \leq i \leq n$, each hold a single input bit to f. Players Q_j, $1 \leq j \leq k$ and players R_1, R_2 hold no input. Fix an arbitrary field F (say, $F = \mathrm{GF}(2)$).

The protocol proceeds as follows. Each player P_i randomly breaks its input into k additive shares, and sends each jth share to all the Q players *except* Q_j. The Q players now apply the time-efficient multi-party communication protocol for f, but instead of sending the answer to the referee they break it into two random shares and send each share to a different R player. Finally, each R player adds up the k values it received and sends the sum to all players. The output is the sum of the two values sent by the R players. □

Remark 5.10 In the computational setting for secure computation, where the information-theoretic privacy requirement is relaxed to hold against computationally-bounded players (under cryptographic assumptions), the answer to the first part of Question 5.8 is affirmative [Yao86, BMR90]. However, the second part of this question is open in this setting as well. Thus, the tradeoff questions in this section are well motivated also when restricting the attention to the computational model.

References

[BF91] Babai, L., and Fortnow, L. Arithmetization: A new method in structural complexity theory. *Computational Complexity 1* (1991), 41–66.

[BGKL03] Babai, L., Gál, A., Kimmel, P. G., and Lokam, S. V. Communication complexity of simultaneous messages. *SIAM Journal of Computing 33*, 1 (2003), 137–166. (Preliminary Version in *12th STACS*, 1995).

[BTY94] Beame, P., Tompa, M., and Yan, P. Communication-space tradeoffs for unrestricted protocols. *SIAM Journal of Computing 23*, 3 (June 1994), 652–661. (Preliminary Version in *31st FOCS*, 1990).

[BFKR90] Beaver, D., Feigenbaum, J., Kilian, J., and Rogaway, P. Security with low communication overhead. In *Proc. 10th Annual International Cryptology Conference (CRYPTO 1990)* (Santa Barbara, California, 11–15 Aug. 1990), A. Menezes and S. A. Vanstone, Eds., vol. 537 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 62–76.

[BMR90] Beaver, D., Micali, S., and Rogaway, P. The round complexity of secure protocols (extended abstract). In *Proc. 22nd ACM Symp. on Theory of Computing* (Baltimore, Maryland, 4–16 May 1990), pp. 503–513.

[BGW88] Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proc. 20th ACM Symp. on Theory of Computing* (Chicago, Illinois, 2–4 May 1988), pp. 1–10.

[CG93] Canetti, R., and Goldreich, O. Bounds on tradeoffs between randomness and communication complexity. *Computational Complexity 3* (1993), 141–167. (Preliminary Version in *31st FOCS*, 1990).

[CFL83] Chandra, A. K., Furst, M. L., and Lipton, R. J. Multi-party protocols. In *Proc. 15th ACM Symp. on Theory of Computing* (Boston, Massachusetts, 25–27 Apr. 1983), pp. 94–99.

[CCD88] Chaum, D., Crépeau, C., and Damgård, I. Multiparty unconditionally secure protocols (extended abstract). In *Proc. 20th ACM Symp. on Theory of Computing* (Chicago, Illinois, 2–4 May 1988), pp. 11–19.

[DGS87] Duris, P., Galil, Z., and Schnitger, G. Lower bounds on communication complexity. *Information and Computation 73*, 1 (Apr. 1987), 1–22.

[GL89] Goldreich, O., and Levin, L. A. A hard-core predicate for all one-way functions. In *Proc. 21st ACM Symp. on Theory of Computing* (Seattle, Washington, 15–17 May 1989), pp. 25–32.

[GMW87] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symp. on Theory of Computing* (New York City, NY, 25–27 May 1987), pp. 218–229.

[HSS93] Håstad, J., Schrift, A. W., and Shamir, A. The discrete logarithm modulo a composite hides $O(n)$ bits. *Journal of Computer and System Sciences 47*, 3 (Dec. 1993), 376–404. (Preliminary Version in *22nd STOC*, 1990).

[LFKN92] Lund, C., Fortnow, L., Karloff, H. J., and Nisan, N. Algebraic methods for interactive proof systems. *Journal of the ACM 39*, 4 (Oct. 1992), 859–868. (Preliminary Version in *31st FOCS*, 1990).

[NW93] Nisan, N., and Wigderson, A. Rounds in communication complexity revisited. *SIAM Journal of Computing 22*, 1 (Feb. 1993), 211–219. (Preliminary Version in *23rd STOC*, 1991).

[PS84] Papadimitriou, C. H., and Sipser, M. Communication complexity. *Journal of Computer and System Sciences 28*, 2 (Apr. 1984), 260–269. (Preliminary Version in *14th STOC*, 1982).

[SS90] Schrift, A. W., and Shamir, A. The discrete log is very discreet. In *Proc. 22nd ACM Symp. on Theory of Computing* (Baltimore, Maryland, 14–16 May 1990), pp. 405–415.

[Yao79] Yao, A. C.-C. Some complexity questions related to distributive computing (preliminary report). In *Proc. 11th ACM Symp. on Theory of Computing* (Atlanta, Georgia, 30 Apr.–2 May 1979), pp. 209–213.

[Yao86] Yao, A. C.-C. How to generate and exchange secrets? (extended abstract). In *Proc. 27th IEEE Symp. on Foundations of Comp. Science* (Toronto, Ontario, Canada, 27–29 Oct. 1986), pp. 162–167.

A Proof of Lemma 3.11

Call an oracle L (C, T)-*bad* if there exists a communication protocol π such that for infinitely many input lengths n the following properties hold.

- π uses $C(n)$ bits of communication;
- π runs in time $T(n)$;
- π succeeds well in predicting the inner product of two random inputs from L_n. Specifically:

$$\Pr_{x,y \in_R L_n}[\pi^L(x, y) = \langle x, y\rangle] > 3/4.$$

By an averaging argument, we may assume without loss of generality that π is deterministic. Furthermore, we first make the simplifying assumption that π queries L only on inputs whose length is equal to its input length. Later we will describe how to remove this assumption.

To prove Lemma 3.11 it suffices to show that for $C(n) = o(n)$ and $T(n) = 2^{o(k(n))}$, a random L is (C, T)-bad with probability 0. We start by showing a procedure for turning any efficient protocol π into an efficient protocol π' which does not make any calls to L. Except with probability 0 over the choice of L, this transformation will maintain the advantage of π in predicting the inner product of two inputs from L.

Definition A.1 *Let M be an oracle TM. We say that M is C-*lucky with respect to L *if there are infinitely many $x \in L$ for which there exists $c \in \{0, 1\}^{C(|x|)}$ such that $M^L(x, c)$ queries L on some input x' such that $x' \neq x$, $|x'| = |x|$, and $x' \in L$.*

Lemma A.2 *Let C, T, k be such that $C(n) + \log(T(n)) + 2k(n) < n/2$, and suppose that M makes at most $T(n)$ calls to its oracle. Then*

$$\Pr_L[M \text{ is } C\text{-lucky with respect to } L] = 0.$$

Proof: Fix some input x of length n and c of length $C(n)$. Each attempt of M to query L succeeds with probability at most $2^{k(n)-n}$ to find x' as above. Taking the union over all $T(n)$ queries, $x \in L_n$ and c, the probability that M queries some x' as above is bounded by $2^{k(n)+C(n)}T(n)2^{k(n)-n} = 2^{-\Omega(n)}$. Thus, the expected number of inputs on which M is lucky is finite. It follows that the probability that L allows M to be lucky on infinitely many x is 0. □ □

Since there are countably many M, we can reverse the order of quantifiers on M and L. In particular, assuming that $C(n), k(n) = o(n)$ we get:

Lemma A.3 *Suppose $C(n) + \log(T(n)) + 2k(n) < n/2$. Then,*

$$\Pr_L[\exists M \text{ such that } M \text{ runs in time } T(n) \text{ and } M \text{ is } C\text{-lucky with respect to } L] = 0.$$

From here on, suppose and $C(n), \log(T(n)), k(n) = o(n)$, so that the conclusion of Lemma A.3 holds. We now suggest the following transformation of a protocol π using time T and communication C into a protocol π' which does not make any calls to L. The protocol π' runs π, simulating oracle calls as follows. Whenever the original input is called the answer is taken to be "yes" and whenever a different input is called the answer is taken to be "no". It is not hard to see that if for some oracle L the outputs of π' and π disagree on infinitely many inputs $(x, y) \in L \times L$ (where $|x| = |y|$), then either Alice's algorithm or Bob's algorithm define M which is lucky with respect to L. Thus, by Lemma A.3, with probability 1 over L the above transformation will maintain the advantage of every protocol π with the given communication and time bounds.

We are now ready to wrap up the proof of Lemma 3.11. Suppose towards contradiction that L is bad with some nonzero probability. It follows from the above that with the same probability over the choice of L, there is a protocol π' predicting the inner product of two random inputs $x, y \in L$ using $T(n)$ time and $C(n)$ communication and without making any calls to L. Since it would be impossible to achieve this advantage if x, y were chosen uniformly from $\{0,1\}^n$, π' can be used to distinguish between the two input distributions with a constant advantage. However, except with probability 0 over the choice of L, there is no such distinguisher running in time $T(n) = 2^{o(k(n))}$.

This gives the desired contradiction, and completes the proof of Lemma 3.11.

Recall, however, that we have made the simplifying assumption that on inputs of length n, the protocol is only allowed to query the oracle on strings of length n. The general case can be similarly handled as follows. Divide oracle queries into "short" (of length at most $k(n)$) and "long". Long queries are easy to simulate using the previous argument. Short queries can be simulated by explicitly "wiring" the answer to all positive instances into π' as an additional advice. Since $k(n) = o(n)$, there are $2^{o(k(n))}$ positive instances, and so the total size of the advice string is also $2^{o(k(n))}$. The proof is completed by noting that even a nonuniform advice of size $2^{o(k(n))}$ does not help distinguish between a random string from L_n and a random string from $\{0,1\}^n$. □