Communication vs. Computation[*]

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

October 18, 2005

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 between communication and time complexity in the following scenarios:

- **Two-party communication complexity.** We exhibit a polynomial-time computable Boolean function that has a low randomized communication complexity, while any communication-efficient protocol for this function requires a super-polynomial amount of local computation. In the case of *deterministic* two-party protocols, we show a similar result relative to a random oracle.
- **Query complexity.** We exhibit a polynomial-time computable Boolean function that can be computed by querying a few bits of its input, but where any such query-efficient scheme requires a super-polynomial amount of computation.
- **Property testing.** We exhibit a polynomial-time decidable property that can be tested (i.e., strings which have the property can be distinguished from ones that are far from the property) by querying a few bits of the input, but where any such query-efficient tester requires a super-polynomial amount of computation.

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.

[*]A preliminary version of this paper appeared in *Proc. 31st International Colloquium of Automata, Languages and Programming (ICALP)*, 2004 [HIK$^+$04].

[†]Toyota Technological Institute, Chicago 60637, USA, E-mail: prahladh@tti-c.org. Research done while the author was at NEC Laboratories America.

[‡]Computer Science Department, Technion, Haifa 32000, Israel, E-mail: yuvali@cs.technion.ac.il. Research supported in part by the Sidney Goldstein Research Fund.

[§]Computer Science Department, Rutgers University, New Jersey 08854, USA, E-mail: jkilian@cs.rutgers.edu. Research done while the author was at NEC Laboratories America.

[¶]Department of Computer Science, Ben-Gurion University, Beer Sheva, 84105, Israel, E-mail: kobbi@cs.bgu.ac.il. 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 $\mathbb{F}$ and let M be a $n \times k$ matrix over $\mathbb{F}$. The columns of M 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:

> When $k > n$, does the function $\mathsf{PS}(M)$ admit a protocol in which (1) each player only sends a single element of $\mathbb{F}$; 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.

Somewhat 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 $\mathbb{F}$ 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 round complexity vs. time complexity 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 Section 1.2). 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 has a protocol with low communication complexity given no restriction on the computation; and (2) there is no protocol for f which simultaneously has low communication and efficient computation. To avoid trivial tradeoffs due to hardness of computing f we also require that f can be efficiently computed given both its inputs, i.e., given no restriction on the communication.

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, unless $P = NP$, there is no protocol that simultaneously uses efficient computation and *optimal* communication (in the sense of minimizing the size of the set of possible messages sent from A to B for the given input instance). Indeed, in such an "optimal" protocol the message sent by A should directly correspond to the equivalence class of g: by the protocol's correctness two non-equivalent formulas cannot induce the same message, and by its optimality two equivalent formulas must induce the same message. Thus, 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, Tompa and Yan [BTY94]. Tradeoffs between randomness and communication were studied by Canetti and Goldreich [CG93].

[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.

1.3 Our Results

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

Randomized communication model. Suppose that there is a UP relation R such that the search problem corresponding to R is not solvable by any probabilistic algorithm running in time $2^{O(T(n))}$. (This would follow from the existence of a one-way permutation secure against a $2^{O(T(n))}$ bounded adversary. In fact, it suffices for our purposes that the permutation be hard to invert in the *worst case.*) 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 $f(x, y)$ computable in polynomial time such that the randomized communication complexity of f is bounded by c. Similarly, let F_c^{poly} be the functions $f(x, y)$ computable in polynomial time such that $f(x, y)$ is computable by two probabilistic 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.[3] 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, given any $\varepsilon > 0$, there exists a language L such that, on inputs of length n, a query scheme with unlimited computation makes n^ε queries while a query scheme with efficient local computation requires $\Omega(n^{1-\varepsilon})$ queries; (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.

Tradeoff questions motivated by secure multi-party computation. 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:

[3]In particular, the lack of efficient equality test prevents a direct translation of our result for the randomized communication model into the deterministic communication model.

- **Arithmetization of Boolean functions.** We consider a possible tradeoff between the *degree* of a polynomial extension and the *time* required to evaluate it. Does every efficiently computable function admit a low-degree extension that can be efficiently evaluated?
- **Multi-party communication.** We consider a generalization of the riddle described above, where instead of computing the product of the row sums one computes a general function f of the row sums. Can the riddle be solved for any efficiently computable function f?
- **Cryptography.** We consider the problem of secure multi-party computation, where several parties wish to evaluate a function f on their joint inputs without revealing their inputs to each other. We suggest a possible tradeoff between the time complexity and communication complexity of secure protocols with a constant number of rounds. Does every efficiently computable f admit a constant-round secure protocol with efficient local computation?

We relate the above questions 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. Hence, the cryptographic application may serve as an additional motivation for studying the other two problems.

1.4 Tradeoffs without assumptions?

An interesting question that arises when discussing such tradeoffs is that of finding the minimal complexity assumptions under which the tradeoffs exist. For instance, the communication vs. computation tradeoff for the randomized communication model is based on the existence of hard UP relations. Can this assumption be weakened? Can such tradeoffs be established assuming only P$\neq$NP, or even unconditionally? If we consider a very simple model such as a 1-round deterministic communication model, then the communication-optimal (1-round) protocol can be made computation-efficient given an oracle to Σ_2. Thus, a tradeoff in this model implies P$\neq$NP. As mentioned in Section 1.1, P$\neq$NP is also a sufficient assumption for establishing a weak tradeoff in this model. The situation in more general models is not clear. In fact, it is not clear that unconditional tradeoff results cannot be established.

Another interesting open problem is to obtain an "explicit" tradeoff in the deterministic communication model (i.e., not relative to a random oracle). The question of obtaining strong tradeoff results of this type (in contrast to the weak tradeoff sketched in Section 1.1) appears to be challenging even in the simple case of 1-round protocols.

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 $f : X \times Y \rightarrow Z$ 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 communication 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 communication cost of a protocol P is the worst case communication cost of P over all inputs (x, y). The (deterministic) communication complexity of f is the minimum communication 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 probability at most $\frac{1}{4}$ on any input (x, y). The probability of error is taken over the internal coin tosses of the protocol. We refer the reader to the book on Communication Complexity by Kushilevitz and Nisan [KN97] for a detailed account of this model.

In general, we will consider a family of functions $\mathcal{F} = \{f_n : n \in \mathbb{Z}^+\}$ and the corresponding family of protocols $\{P_n : n \in \mathbb{Z}^+\}$ such that protocol P_n computes the function f_n, for each $n \in \mathbb{Z}^+$. The communication complexity of a family of functions $\mathcal{F}$ is the function $C : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that $C(n)$ is the communication complexity of the function f_n. For ease of writing, henceforth whenever we say function (protocol), we really mean a "family of functions" ("family of protocols"), one for each $n \in \mathbb{Z}^+$.

2.2 Tradeoffs

We now describe formally our tradeoff problem in the two-party randomized 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 randomized 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 probabilistic polynomial time for local computation requires almost a 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, for some constant 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 denote the set $\{x \in \Sigma^* : \exists w, (x, w) \in R\}$ by L_R. Thus, $w(x)$ is defined if and only if $x \in L_R$.

We will assume the existence of UP relations for which the corresponding search problem is very hard. Such an assumption is standard in cryptography. In particular, 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 Communication 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 for some constant k. Consider the 2-player (Alice and Bob) Boolean function $f_R : \{0,1\}^{n+n^k} \times \{0,1\}^{n^k} \rightarrow \{0,1\}$.*

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

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 relation. Then, the predicate f_R has the following properties.*

1. *f_R is computable in polynomial time.*
2. *There exists a 1-round 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 1-round 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[4] 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.

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 $\mathrm{poly}(n, 2^{b(n)}) \cdot T_A(n)$.*

[4]Recall that there exists a 1-round protocol for equality with randomized communication complexity at most $O(\log n)$ [KN97].

Proof:

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

To begin with, let us assume that the input x is in the set L_R. Note that there might not exist an efficient algorithm for deciding if $x \in L_R$ unless NP=P. Our goal is to relate the search problem of computing w (if it exists), 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, given x and z, computes $\langle z,w\rangle$ with some advantage over random guessing. Given such an oracle, we can then use the Goldreich-Levin reconstruction algorithm [GL89] to compute a small number of candidates for w. More precisely, we create a "small" set of oracles such that one of the oracles computes $\langle z,w\rangle$ with some nontrivial advantage. We then try each oracle by exhaustive search, and use the fact that we can recognize the correct w.

Converting protocols into oracles

Let Π be a randomized protocol for f_R with $b(n)$ communication. Let $\mathcal{T}$ be a transcript of Π. For simplicity, we assume Alice outputs $f_R((x,z),w)$ as her final bit; this convention increases the size of the transcript by at most 2 bits. Thus, $\mathcal{T}$ includes a "guess" for $\langle z,w\rangle$ as the last bit. Using $\mathcal{T}$, 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.

We are interesting in analyzing the advantage of the oracle above in guessing $\langle z,w\rangle$. To do that, we define our notation for the advantage of Π and $A_{\mathcal{T}}$ in guessing $\langle z,w\rangle$.

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

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

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

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

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, for every $x \in L_R$

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

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 $x \in \{0,1\}^n \cap L_R$, 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 $\text{ADV}(A_{\mathcal{T}}, x)$ by $\text{ADV}(\Pi, x, \mathcal{T})$.

Proposition 3.6 *For every $x \in L_R$, 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), 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

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

Solving the search problem

As shown above, there exists a transcript $\mathcal{T}^*$ such $A_{\mathcal{T}^*}$ has a good advantage in guessing $\langle z, w \rangle$. We now show that we can solve the search problem using $A_{\mathcal{T}^*}$. Set $\varepsilon = \frac{1}{2^{2b(n)+1}}$. 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 $\mathrm{poly}(n, 1/\varepsilon)$ 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 $2^{2b} \cdot \mathrm{poly}(n, 2^{2b+1}) \cdot T_A(n) = \mathrm{poly}(n, 2^b) \cdot T_A(n)$. So far we have assumed $x \in L_R$. However, since we have an efficient algorithm to recognize the correct w (if it exists), we need not assume that $x \in L_R$. One may as well run the above algorithm on any x (not necessarily those in L_R) and finally check if the candidate w is the correct one (if it exists). This proves Lemma 3.4. $\square$

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, $\mathrm{poly}(n, 2^b) \cdot T_A(n) \geq 2^{\Omega(T(n))}$ and hence $b(n) = \Omega(T(n))$ since $T_A(n)$ is polynomially bounded in n. $\square$

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 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. By this we mean that relative to essentially all oracles of a certain class, we can obtain a provable tradeoff. We stress that one cannot use the 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 $L \subseteq \{0,1\}^*$ be any language. Consider the 2-player (Alice and Bob) Boolean function f_L, defined on inputs of length n as follows.*

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

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

Theorem 3.9 *Let $k(n)$ be any time-constructible function such that $k(n) = \omega(\log n)$, $k(n) = o(n)$, and $n - k(n)$ is monotone non-decreasing. Let L be a random $2^{k(n)}$-sparse language, namely a random language containing $2^{k(n)}$ strings of length n. Then, with probability 1 over the 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 1-round deterministic communication protocol with $k(n) + 1$ bits of communication.*
3. *There is no sublinear-communication protocol (deterministic or randomized) in which Alice and Bob are bounded to polynomial time (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. $\square$

In the remainder of this section we establish Property 3, which is a simple corollary of the following more general lemma.

Lemma 3.11 *Let C, T, k be such that $C(n) = o(n)$, $T(n) = 2^{o(k(n))}$, and $k(n)$ as in Theorem 3.9. Let L be a random $2^{k(n)}$-sparse language. Then, with probability 1 over the choice of L there is no communication protocol for f_L in which Alice and Bob run in time $T(n)$ and exchange $C(n)$ 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.*

Proof overview. The proof of Lemma 3.11 proceeds as follows. Suppose that we are given 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 since L is *sparse*, a time-bounded protocol is unlikely to query L on any input $q \in L$ 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 time-efficient 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* language, this leads to a contradiction.

Proof of Lemma 3.11. We refer to a communication protocol π as being (C, T)-*bounded* if it uses at most $C(n)$ communication bits and its running time is at most $T(n)$ in the worst case (when running on inputs of length n). Furthermore, if π is an oracle-aided protocol, then these time and communication bounds should hold relative to all oracles.[5] Similarly, we say that an oracle-aided TM M is T-bounded, if its running time is bounded by $T(n)$ regardless of the given oracle.

[5]As long as C, T are "well-behaved" in the sense that they can be computed in time $O(T(n))$, any protocol π which is (C, T)-bounded relative to a *specific* oracle L can be easily converted to a $(C, O(T))$-bounded protocol $\hat{\pi}$, such that π^L and $\hat{\pi}^L$ have the same outputs. Throughout this section it is assumed that the functions C, T are well behaved.

Definition 3.12 (**(C,T)-bad oracle**) *A language $L \subseteq \{0,1\}^*$ is called* (C,T)-bad *if there exists a (C,T)-bounded protocol π, having oracle access to L, such that for infinitely many input lengths n,*

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

That is, π^L "succeeds well" in predicting the inner product of two random inputs from L of length n.

Note that if the requirements of Definition 3.12 are satisfied by a randomized protocol π then, by an averaging argument, there is also a *deterministic* protocol which satisfies these requirements using a nonuniform advice of length $T(n)$. (For each input length n, the advice will consist of a sequence of coin-tosses which maximizes the success probability.) We may thus assume from now on that the protocol π is deterministic and uses advice of size $T(n)$.

To prove Lemma 3.11 it suffices to show the following.

Claim 3.13 *Let C, T, k be as in Lemma 3.11. Then a random $2^{k(n)}$-sparse oracle L is (C,T)-bad with probability 0.*

We start by showing a procedure for turning a (C,T)-bounded oracle-aided protocol π and a $2^{k(n)}$-sparse language L into a (C,T)-bounded 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^L in predicting the inner product of two inputs from L. The protocol π' runs π, simulating oracle calls as follows.

- If the oracle query is "short", namely of size smaller than $3k(n)$, the answer will be wired into π'. That is, π' will include a nonuniform advice of size $(3k(n))^2 \cdot 2^{k(3k(n))} = 2^{o(k(n))}$ containing all short oracle queries to which the answer is "yes". (Together with the advice of size $T(n)$ used to derandomize π, the total advice size is still $2^{o(k(n))}$.)
- Long oracle queries, of size $3k(n)$ or more, are handled as follows. Whenever a party needs to make a long query different from its input, it takes the answer to be "no". Whenever the query string is equal to the input the answer is taken to be "yes".

We turn to analyze the success of π' on inputs from $L \times L$. Short queries pose no problem since, by the definition of π', they are simulated correctly. We argue that long queries are very unlikely to be simulated incorrectly. That is, when running π^L on $(x,y) \in L \times L$, each party is unlikely to make a long query from L which differs from its local input. This is formalized below.

Definition 3.14 (Offending queries) *Let π be a deterministic oracle-aided protocol and $L \subseteq \{0,1\}^*$ be a language. We say that π^L makes an* offending query *on input (x,y) if some party queries the oracle on a string $q \in \{0,1\}^*$ such that $q \in L$, $|q| \geq 3k(n)$, and q is different from its local input.*

Lemma 3.15 *Let C, T, k be as in Lemma 3.11. Let π be a deterministic, (C,T)-bounded oracle-aided protocol. Then, with probability 1 over the choice of a random $2^{k(n)}$-sparse L, there are only finitely many $(x,y) \in L \times L$ such that $|x| = |y|$ and π^L makes an offending query on input (x,y).*

Proof: It suffices to show that the expected number of inputs $(x,y) \in L \times L$ on which offending queries are made is finite, where the expectation is taken over the choice of L. For this, we separate offending queries into two types. The first type includes queries whose length is equal to the input length, and the second type includes the remaining queries.

The main intuition for analyzing offending queries of the first type is that the bound on the communication complexity prevents the parties from guessing each other's input, and the bound on the time complexity prevents them from finding a string in L via exhaustive search. To formalize this intuition, consider the view of one of the parties (say Alice) on some input $x \in L$ of length n. Each possible communication transcript c, of length $C(n)$, uniquely determines the queries made by Alice when running π^L on input x (independently of Bob's input y). Let $Q_{L,x,c}$ denote this set of queries. Consider the experiment of picking L at random from the set of $2^{k(n)}$-sparse languages, and let B_n denote the event that $Q_{L,x,c}$ includes an offending query of the first type for *some* $x \in L_n$ and $c \in \{0,1\}^{C(n)}$.

We would like to argue that B_n occurs with exponentially small probability. For each fixed x, c, the event that $Q_{L,x,c}$ includes an offending query of the first type (of length n) occurs with at most $T(n) \cdot 2^{k(n)-n}$ probability. (This is analogous to the event that 2^k balls placed randomly in 2^n distinct bins will hit some fixed subset of T bins.) By a union bound, the probability of the latter event occurring for *some* c is bounded by $2^{C(n)} \cdot T(n) \cdot 2^{k(n)-n}$. We would have liked to use a similar union-bound argument over all $x \in L$; however, since the set of x over which we take the union depends on the choice of L we cannot apply such an argument directly. To get around this difficulty, augment the previous experiment by picking a random labeling σ of the elements in L; that is, σ_n is a random bijection such that $\sigma_n([2^{k(n)}]) = L_n$. Defining $Q'_{L,\sigma,n,i,c} = Q_{L,\sigma_n(i),c}$, it follows by symmetry that for any fixed n, i, c the event that $Q'_{L,\sigma,n,i,c}$ includes an offending query of the first type also occurs with at most $T(n) \cdot 2^{k(n)-n}$ probability. (This is because the knowledge of $\sigma_n(i)$ gives no information about the other strings in L_n; note that this would no longer be true if σ_n were fixed to be, say, the lexicographic ordering of L_n.) Now we can take the union over all $i \in [2^{k(n)}]$ and $c \in \{0,1\}^{C(n)}$ and conclude that

$$\Pr[B_n] \leq 2^{C(n)+k(n)} \cdot T(n) \cdot 2^{k(n)-n} = 2^{-\Omega(n)}.$$

It follows that the expected number of inputs (x, y) on which Alice makes an offending query of the first type is finite. Since the same holds for Bob, the expected number of inputs on which *some* party makes an offending query of the first type is also finite, as required.

We turn to analyzing offending queries of the second type. This case is somewhat easier, since we no longer need to worry about the parties communicating their inputs to each other. In fact, the current analysis does not depend on $C(n)$ and only relies on $T(n)$ being sufficiently small with respect to $k(n)$. As before, consider the experiment of picking L at random along with a random labeling σ of its elements, and denote by $Q'_{L,\sigma,n,i,j}$ the set of queries made by the two parties on input $(\sigma_n(i), \sigma_n(j))$. For any fixed n, i, j, the probability that these queries include an offending query of the second type is bounded by $T(n) \cdot 2^{k(3k(n))-3k(n)} = 2^{-3k(n)+o(k(n))}$. This follows from the facts that (1) the length of offending queries is (by definition) at least $3k(n)$; (2) $n - k(n)$ is monotone non-decreasing (thus it suffices to consider queries of the minimal length); and (3) the inputs $\sigma_n(i), \sigma_n(j)$ do not give any information about words in L that are not of length n. Hence, making an offending query of the second type is at least as hard as finding a non-empty bin in $T(n)$ trials, where there are $2^{k(3k(n))}$ balls randomly placed in $2^{3k(n)}$ distinct bins.

Taking the union over all $i, j \in [2^{k(n)}]$, the probability that an offending query of the second type is made on some input $(x, y) \in L_n \times L_n$ is bounded by $2^{-k(n)+o(k(n))}$. Since $k(n) = \omega(\log n)$, the expected number of inputs on which offending queries of the second type are made is finite.

Altogether, the expected number of inputs on which an offending query of either type is made is finite. Hence, the probability (over L) that π^L makes offending queries on infinitely many inputs is 0. □

Lemma 3.15 implies that for any fixed π, the transformed protocol π' will produce the same output as π^L on all but finitely many inputs, except with probability 0 over the choice of L. Since there are only countably many (uniform)[6] protocols π, we get the following stronger corollary.

Lemma 3.16 *Let C, T, k be as in Lemma 3.11. Then, with probability* 1 *over the choice of a random $2^{k(n)}$-sparse L, the following holds. For* every *protocol π (possibly depending on L), its transformed protocol π' produces the same output as π^L on all but finitely many $(x, y) \in L \times L$.*

We are now ready to wrap up the proof of Claim 3.13. Suppose towards a contradiction that Claim 3.13 does not hold. That is, for some $C(n) = o(n)$ and $T(n) = 2^{o(k(n))}$, a random $2^{k(n)}$-sparse L is (C, T)-bad with some nonzero probability p. From Lemma 3.16 we may conclude that, with the same probability over the choice of L, there is a (C, T)-bounded protocol π' predicting the inner product of two random inputs $x, y \in L$ without making any calls to L. That is, for a fraction $p > 0$ of languages L there is (C, T)-bounded π' such that for infinitely many n,

$$\Pr_{x,y \in_R L_n}[\pi'(x, y) = \langle x, y\rangle] > 3/4. \tag{8}$$

By the hardness of the inner product function [CG88], for any protocol π' with communication complexity $C(n) = o(n)$ we have:

$$\Pr_{x,y \in_R \{0,1\}^n}[\pi'(x, y) = \langle x, y\rangle] < 1/2 + n^{-\omega(1)}. \tag{9}$$

Combining Eq. (8) and Eq. (9), one can use π' to get a T-bounded *distinguisher* between a pair of inputs (x, y) drawn at random from $\{0, 1\}^n$ and a pair of inputs drawn at random from L_n. Specifically, for a fraction $p > 0$ of the $2^{k(n)}$-sparse languages L we get a constant distinguishing advantage. However, by a standard counting argument (cf. [Gol01], Section 3.2.2) such a distinguisher does not exist when $T(n) = 2^{o(k(n))}$, except with probability 0 over the choice of L, even if a nonuniform advice of length $2^{o(k(n))}$ is allowed. This completes the proof of Claim 3.13, from which Lemma 3.11 immediately follows. □

4 Tradeoffs in Related Models

4.1 Query vs. Time Complexity for Decision Procedures

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 will also consider the computationally bounded analog of this measure, where D is restricted to run in probabilistic polynomial time.

Definition 4.1 *We define $lsb_\ell(x)$ to be ℓ least significant bits of x. We say that a permutation $p : \{0, 1\}^m \to \{0, 1\}^m$ is $\ell(m)$-lsb hard if p can be computed by a polynomial time algorithm but no*

[6] Recall that we introduced uniformity in order to derandomize a randomized protocol π; however, it suffices to count over the protocols π we started with, of which there are countably many.

probabilistic polynomial-time procedure A can compute $lsb_{l(m)}(p^{-1}(x))$ with probability non-negligibly greater than $2^{-\ell(m)}$, where the probability is over x chosen uniformly from $\{0,1\}^m$ and the internal coin tosses of A.

We note that $C \log m$-lsb hard permutations exist based on the hardness of computing discrete logarithms over composite integers for every constant $C > 1$ [SS90, HSS93]. It is to be noted that [SS90, HSS93] prove stronger results regarding the simultaneous hardness of bits. We mention only the $O(\log n)$ simultaneous hardness as this is what is required for our purposes.

Theorem 4.2 *Let $p : \{0,1\}^m \to \{0,1\}^m$ be a permutation that is $\ell(m)$-lsb hard for $\ell(m) = C \log m$ for some constant $C > 1$. Then there exists a predicate*

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

with the following properties:

1. *C_p is computable in polynomial time.*
2. *The query complexity of C_p is at most $2m$.*
3. *For any constant $\alpha < 1$, there is no probabilistic polynomial-time bounded decision procedure Q that can compute C_p after querying only $2^{\alpha\ell(m)}$ bits.*

Proof: For notational simplicity, we write ℓ instead of $\ell(m)$. We abuse notation and use i both as a numerical value and as an ℓ-bit string. The predicate C_p is defined as follows:

$$C_p(y, x_1, \ldots, x_{2^\ell}) = \begin{cases} 1 & \text{if } p(x_i) = y \text{ and } lsb_\ell(x_i) = i \text{ for some } 1 \le i \le 2^\ell \\ 0 & \text{otherwise} \end{cases}$$

In the definition, y and x_i are in $\{0,1\}^m$. Property 1 is clearly satisfied as C_p is computable in polynomial time e.g. by running over all the (polynomially-many) possible values of i. To see that property 2 is satisfied, consider a computationally unbounded decision procedure for C_p that (i) queries y (which is m bits long), (ii) inverts p to compute $x = p^{-1}(y)$ and $i = lsb_\ell(x)$, (iii) queries x_i (which is m bits long), and accepts iff $x_i = x$.

To conclude the proof, we show that the existence of a probabilistic polynomial-time bounded decision procedure Q for C_p with query complexity $2^{\alpha\ell}$ contradicts our assumption that p is ℓ-lsb hard. Given Q, we construct a polynomial-time algorithm G for guessing $lsb_\ell(x)$ for input $y = p(x)$ as follows:

1. Choose $x_1, \ldots, x_{2^\ell}$ uniformly at random from $\{0,1\}^n$.
2. Run Q on input $(y, x_1, \ldots, x_{2^\ell})$. Let $I = \{i : Q \text{ queries at least one bit of } x_i\}$.
3. Randomly choose an index $i \in I$. Output i as an ℓ-bit string.

Lemma 4.3 *G computes $lsb_\ell(x)$ with success probability at least $\frac{1}{2^{\alpha\ell+1}}$.*

Proof: Fix an input $t = (y, x_1, \ldots, x_{2^\ell})$ for Q such that $C_p(t) = 1$. Let $p^{-1}(y) = x$ and $i = lsb_l(x)$. Then, by definition, $x_i = x$. Therefore, Q on input t outputs 1 with probability at least 3/4 over its internal coin tosses. We first show that with probability at least 1/2, the decision procedure Q both outputs 1 and the set I (constructed in Step. 2 above) contains the index i.

Suppose this were not the case. Then with probability greater than 1/4, the decision procedure Q outputs 1 and the set I does not contain the index i. This would imply that the decision procedure Q outputs 1 with probability greater than 1/4 on the instance t', $C_p(t') = 0$, obtained from t by replacing $x_i = x$ in t by $x' \neq x$. This contradicts the definition of the decision procedure Q. Hence, with probability at least 1/2, the set I contains the index i. Now the lemma follows from the fact that if $i \in I$, G outputs it with probability at least $1/|I|$ and $|I| \leq 2^{\alpha\ell}$. □

For $l = O(\log m)$, G computes the $lsb_l(x)$ with success probability non-negligibly greater than 2^{-l}, a contradiction. This proves property 3 completing the proof of the Theorem. □

We now observe that the tradeoff result for query complexity stated in Section 1.3 follows from Theorem 4.2 and the fact that for any constant $C > 1$, $C \log n$-lsb hard permutations exist [SS90, HSS93] as stated before.

4.2 Query vs. Time Complexity 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 (in Hamming distance) 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.

We will also consider a computationally bounded analog of this measure, where the tester T is restricted to run in probabilistic polynomial time. We now present a predicate that exemplifies a computation-communication tradeoff in this model:

Theorem 4.4 *Let L be any language such that $L \in NP$ and $L \notin BPP$. Fix any $\varepsilon > 0$. Fix any $c > 1$ and let $r = m^c$. Then, there exists a predicate*

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

with the following properties:

1. *P_L is computable in polynomial time.*
2. *The ε-testing complexity of P_L is $O(m)$.*
3. *No probabilistic polynomial-time bounded property ε-tester T can compute P_L querying less than r 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 $R_L(x, w)$ be the NP relation corresponding to L. Without loss of generality, let us assume that $|x| = |w|$ and denote it m. Let $C : \{0,1\}^m \to \{0,1\}^{rm}$ be an error correcting code with distance greater than $2\varepsilon rm$. For $y \in \{0,1\}^{rm}$ and $w_1, \ldots, w_r \in \{0,1\}^m$, we define P_L as follows:

$$P_L(y, w_1, \ldots, w_r) = \begin{cases} 1 & \text{if } y = C(x), x \in L \text{ and } R_L(x, w_1 \oplus \cdots \oplus w_r) \text{ holds} \\ 0 & \text{otherwise} \end{cases}$$

That is, $P_L(y, w_1, \ldots, w_r)$ is 1 if and only if y encodes an string x in L and $w_1, \ldots, w_r$ compose to give a witness for $x \in L$. Property 1 is satisfied because P_L is computable in polynomial time using a procedure that decodes y to get x and then checks that $w_1, \ldots, w_r$ do compose to give a witness $w = w_1 \oplus \cdots \oplus w_r$ for the membership of x in L.

To check that Property 2 is satisfied, consider the following (computationally unbounded) tester:

1. Probe $\frac{m}{4\varepsilon}$ random locations of y.
2. Accept if y agrees with $C(x)$ for some $x \in L$. If not, reject.

It is easy to see that the tester outputs 1 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. $m \ll \varepsilon rm$ bits) to get an instance that satisfies P_L. Therefore, instances that are ε-far from an instance that satisfies P_L must contain y that is ε-far from any $y = C(x)$ for $x \in L$. Hence, our computationally unbounded tester decides the property without looking at $w_1, \ldots, w_r$. The tester probes bits of y and accepts only if they agree with some codeword $C(x)$ for $x \in L$. Since the probability that the bits of y agrees with a fixed codeword $C(x)$ is at most $(1-\varepsilon)^{\frac{m}{4\varepsilon}}$,

$$\begin{aligned} \Pr[\text{The tester accepts}] &\leq 2^m(1-\varepsilon)^{\frac{m}{4\varepsilon}} \\ &\leq \frac{1}{4} \end{aligned}$$

To conclude the proof, suppose that a probabilistic polynomial time bounded tester T exists that queries less than r 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 $w_1, \ldots, w_r$ in $\{0,1\}^m$.
3. Output $T(y, w_1, \ldots, w_r)$.

Since T reads less than r bits of the input, it follows that it does not learn even a single bit of w from probing $w_1, \ldots, w_r$. More formally, we show that from T's point of view the following two distributions are indistinguishable for any w: (a) $(w_1, \ldots, w_r)$ where each of the w_i are chosen independently from the uniform distribution and (b) $(w_1, \ldots, w_r)$ where all the w_i but for w_r are chosen independently from the uniform distribution and w_r is set to $w \oplus \bigoplus_{i=1}^{r-1} w_i$. It suffices to prove this statement for deterministic (possibly adaptive) testers since the behavior of a randomized tester can be expressed as a convex combination of deterministic testers. However this is trivially satisfied for deterministic testers T, since T reads less than r bits of $(w_1, \ldots, w_{r-1})$. Therefore, the algorithm D has the same probability of success as T. This implies that L is in BPP, a contradiction. This proves property 3. □

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 as in the case of query complexity.

5 Tradeoff Questions Motivated by Secure Multi-Party Computation

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, multi-party communication, and the cryptographic problem of secure multi-party computation. 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. Hence, the cryptographic application may serve as an additional motivation for studying the other two.

5.1 Time vs. Degree in Arithmetization

Arithmetization of Boolean functions has proved to be a very powerful tool in complexity theory [LFKN92, BF91]. Let R be a commutative ring with identity. A function $\hat{f} : R^n \to R$ is said to *extend* a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if $\hat{f}$ and f agree on $\{0,1\}^n$; that is, if for every $x \in \{0,1\}^n$ we have $\hat{f}(x) = f(x)$. We refer to $\hat{f}$ as a *degree-d extension* of f over R if it can be computed by an n-variate polynomial $p(X_1, \ldots, X_n)$ over R, whose degree *in each variable* is bounded by d. The polynomial p will be referred to as a *polynomial representation* of f. In typical applications of polynomial representations in complexity theory, R is taken to be either the ring of integers or a finite field. Note that a representation over the integers gives rise to a representation of the same degree over an arbitrary finite field by reducing the coefficients modulo the characteristic of the field.

We consider the computational complexity of evaluating a low-degree representation at a given point in R^n. Specifically, given a function f and a degree bound d, how easy is the "computationally easiest" extension meeting the degree constraint?

In the case $d = 1$, any function f admits a unique extension over R, called the *multi-linear* extension, which can be computed in EXPTIME^{f,R}. But if the function f is "easy" (say, in AC^0) can anything better be said about the computational complexity of its multi-linear extension? Also, if we relax the degree bound to $d > 1$, does this allow for more computationally 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 complexity measure and complexity class.

Definition 5.1 *Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, a bound d on the degree (in each variable) and a prime power q, define $\mathsf{TD}_{d,q}(f)$ to be the minimal integer s such that there is a size-s algebraic circuit (over the basis $\{+, *\}$) evaluating some degree-d extension of f over $F = \mathrm{GF}(q)$. Given functions $d(n), s(n)$, and $q(n)$ (where q returns a prime power), define the class $\mathcal{TD}_{d,s,q}$ to be the class of functions $f : \{0,1\}^* \to \{0,1\}$ such that for every n we have $\mathsf{TD}_{d(n),q(n)}(f_n) \le s(n)$, and furthermore a corresponding circuit of size $s(n)$ (including a description of $q(n)$) can be uniformly generated in time* $\mathrm{poly}(s(n))$.

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 the formula size.

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

Proof: A degree-ℓ extension as required is obtained by the straightforward arithmetization φ. Specifically, recursively define a polynomial $p_\varphi(X_1, \ldots, X_n)$ representing the formula $\varphi(x_1, \ldots, x_n)$ as follows: If $\varphi = x_i$ then $p_\varphi = X_i$; if $\varphi = \neg\varphi'$ then $p_\varphi = 1 - p_{\varphi'}$; if $\varphi = \varphi_1 \wedge \varphi_2$ then $p_\varphi = p_{\varphi_1} \cdot p_{\varphi_2}$; and if $\varphi = \varphi_1 \vee \varphi_2$ then $p_\varphi = 1 - (1 - p_{\varphi_1})(1 - p_{\varphi_2})$. It is easy to verify that p_φ indeed represents φ, the degree of p_φ in each variable (in fact, even the total degree) is bounded by the size of φ, and p_φ can be computed by an arithmetic circuit (in fact, even a formula) of size $O(\ell)$. $\square$

Note, however, that if φ is a *circuit* (rather than a formula), then the degree of the polynomial p_φ produced by the above arithmetization method might become exponential in $|\varphi|$.

Claim 5.2 immediately gives the following:

Corollary 5.3 *For every $f \in \mathrm{NC}^1$ there is a polynomial $d(n)$ such that for all (efficiently computable) $q(n)$ we have $f \in \mathcal{TD}_{d,d,q}$.*

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

Claim 5.4 *There is an AC^0 function f such that if $f \in \mathcal{TD}_{1,s(n),q(n)}$ for some polynomial $s(n)$ and $q(n) > 2$ then $\mathrm{NP} \subseteq \mathrm{BPP}$.*

Proof: Let f be a universal function for 3CNF formulas. Specifically, f has $m = \binom{2n}{3}$ variables $g = (g_1, \ldots, g_m)$, which represent a 3CNF formula on n variables by indicating which clauses are included in the formula, and n variables $z = (z_1, \ldots, z_n)$, representing an assignment to the formula. The output of $f(g; z)$ is $g(z)$, the value of the formula represented by g on the assignment represented by z. The function f can be computed in AC^0.

Let $p(X_1, \ldots, X_k)$ denote the (unique) multi-linear extension of f, where $k = m + n$. The polynomial p can be written as

$$p(X_1, \ldots, X_k) = \sum_{a: f(a)=1} \prod_{i=1}^{k} L_{a_i}(X_i) \tag{10}$$

where $L_b(X_i)$ is defined as X_i if $b = 1$ and as $1 - X_i$ if $b = 0$.

Using a polynomial-size arithmetic circuit computing p over $F = \mathrm{GF}(q)$, one can efficiently solve the unique-3SAT problem, i.e. distinguish between an unsatisfiable 3CNF formula φ and one which is satisfied by exactly one assignment. This is done by computing $p(\varphi; \alpha, \alpha, \ldots, \alpha)$ where α is an arbitrary element of $F \setminus \{0, 1\}$, and comparing the result to 0. Indeed, in the multi-linear extension (10), when restricted to φ, each nonzero term corresponds to a satisfying assignment of φ. Such a term evaluates to $\alpha^w (1 - \alpha)^{(n-w)} \neq 0$, where w is the Hamming weight of the satisfying assignment. The claim then follows from the fact that efficiently solving Unique-3SAT implies $\mathrm{NP} \subseteq \mathrm{BPP}$ [VV86]. □

There are various questions one could ask regarding time-degree tradeoffs. In particular, one could try to improve the degree bounds ℓ and 1 in Claims 5.2 and 5.4 respectively, ideally closing the gap between the two. For concreteness, we would like to highlight the following special cases of the general question.

Question 5.5 *Does every $f \in \mathrm{P}$ admit polynomials $d(n), s(n)$ such that $f \in \mathcal{TD}_{s,d,q}$ for every efficiently computable and polynomially bounded*[7] *$q(n)$? Does the above hold with $d(n) = O(n^c)$, for some constant c independent of f?*

[7]Note that when $q(n) < d(n)$, the degree restriction becomes trivial. On the other hand, we do not want $q(n)$ to be too large, as this would make a positive answer to the question very unlikely. Indeed, if q is exponential in the size of the minimal circuit C computing a degree-d extension $\hat{f}$ over $\mathrm{GF}(q)$, then the degree of C, viewed as a formal polynomial, is also bounded by d. (This is not necessarily true when q is smaller.) It follows from [VSBR83] that if $\hat{f}$ is computed by a polynomial-size arithmetic circuit having a polynomially bounded degree, then $\hat{f}$ is in arithmetic NC^2.

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. For each row i, $1 \leq i \leq n$, define a degree-1 polynomial p_i over F' by $p_i(y) = \sum_{m=1}^{k} M_{im}(1 - y/\alpha_m)$. Define a degree-$n$ polynomial $p(y) = \Pi_{i=1}^{n} p_i(y)$. It is easy to verify that: (1) $p(0) = \mathsf{PS}(M)$ and (2) $p(\alpha_j)$ does not depend on the jth column of M, and thus can be locally evaluated by player j. Since $k > n$ and $\deg p \leq n$, the value $\mathsf{PS}(M) = p(0)$ can be interpolated from the k values $p(\alpha_j)$. That is, we can write $\mathsf{PS}(M) = \sum_{j=1}^{k} \lambda_j p(\alpha_j)$ for some fixed coefficients $\lambda_j \in F'$. Each player j computes $\lambda_j p(\alpha_j)$ (which is an element of F'), projects it down to the original field F using a field homomorphism $h : F' \rightarrow F$, and sends the result to the referee.

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

Thus, we have shown:

Theorem 5.6 *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. This raises the following questions (formulated in correspondence to Question 5.5):

Question 5.7 *Can the generalized communication problem be solved for any* $f \in FP$ *using* $k =$ poly(n) *players, efficient communication (a single field element per player), and polynomial-time computation? Can this be achieved with* k *being independent of the (polynomial) time complexity of* f*?*

Claim 5.8 *A positive answer to Question 5.5 implies a positive answer to Question 5.7.*

Proof: The solution to the riddle can be generalized to an efficient protocol for evaluating any (efficiently computable) degree-d polynomial p in the row sums, using $k = dn + 1$ players. (The only required modification is that player j now applies p to its n intermediate outputs $P_{i,j}$ rather than multiply them.) The claim follows by letting p be the efficient polynomial representation guaranteed by Question 5.5, over an extension field of size $q > k$. □

5.3 Time vs. Communication and Rounds in Cryptography

In the problem of secure multi-party 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). We also assume for simplicity that all players follow the prescribed protocol, though a similar discussion applies to the case of malicious players as well.

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 securely computed by $k = n/\log n$ players[8] (or more) in a constant number of rounds using $\mathrm{poly}(n)$ 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 5.9 *Can any polynomial-time computable $f : \{0,1\}^n \to \{0,1\}$ be securely 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 5.5, one can in fact make the following stronger claim.

Claim 5.10 *Any solution to Question 5.7 in which the referee outputs the sum of the k field elements it receives implies a positive answer to Question 5.9.*

Proof: Let k be the (polynomial) number of players guaranteed by the solution to Question 5.7, and fix an arbitrary finite field F (say, $F = \mathrm{GF}(2)$). We describe a secure protocol for f involving $n + 3k + 2$ players, which are partitioned into 3 disjoint sets. Players P_i, $1 \le i \le n$, each hold a single input bit to f. Players Q_j, $1 \le j \le k$ and players R_1, R_2 hold no input.

The protocol proceeds as follows. Each player P_i randomly breaks each of its inputs x_i into k additive shares $r_1, \ldots, r_k$ (i.e., the r_j are random subject to $\sum r_j = x_i$ where summation is taken over F), and sends each share r_j 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 additive 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. □

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

Remark 5.11 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 5.9 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.

[CG88] Chor, B., and Goldreich, O. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal of Computing 17*, 2 (Apr. 1988), 230–261. (Preliminary Version in *26th FOCS*, 1985).

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

[Gol01] Goldreich, O. *The Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, Cambridge, U.K., 2001.

[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.

[HIK$^+$04] Harsha, P., Ishai, Y., Kilian, J., Nissim, K., and Venkatesh, S. Communication vs. computation. In *Proc. 31st International Colloquium of Automata, Languages and Programming (ICALP)* (Turku, Finland, 12–16 July 2004), J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, Eds., vol. 3142 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 745–756.

[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).

[KN97] Kushilevitz, E., and Nisan, N. *Communication Complexity.* Cambridge University Press, 1997.

[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.

[VSBR83] Valiant, L. G., Skyum, S., Berkowitz, S., and Rackoff, C. Fast parallel computation of polynomials using few processors. *SIAM Journal of Computing 12*, 4 (Nov. 1983), 641–644.

[VV86] Valiant, L. G., and Vazirani, V. V. NP is as easy as detecting unique solutions. *Theoretical Computer Science 47*, 3 (1986), 85–93. (Preliminary Version in *17th STOC*, 1985).

[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.