VERIFIABLE STREAM COMPUTATION AND ARTHUR–MERLIN COMMUNICATION[*]

AMIT CHAKRABARTI[†], GRAHAM CORMODE[‡], ANDREW MCGREGOR[§], JUSTIN THALER[¶], AND SURESH VENKATASUBRAMANIAN[‖]

Abstract.
In the setting of streaming interactive proofs (SIPs), a client (verifier) needs to compute a given function on a massive stream of data, arriving online, but is unable to store even a small fraction of the data. It outsources the processing to a third party service (prover), but is unwilling to blindly trust answers returned by this service. Thus, the service cannot simply supply the desired answer; it must *convince* the verifier of its correctness via a short interaction after the stream has been seen.

In this work we study "barely interactive" SIPs. Specifically, we show that one or two rounds of interaction suffice to solve several query problems—including Index, Median, Nearest Neighbor Search, Pattern Matching, and Range Counting—with polylogarithmic space and communication costs. Such efficiency with $O(1)$ rounds of interaction was thought to be impossible based on previous work.

On the other hand, we initiate a formal study of the limitations of constant-round SIPs by introducing a new hierarchy of communication models called Online Interactive Proofs (OIPs). The online nature of these models is analogous to the streaming restriction placed upon the verifier in a SIP. We give upper and lower bounds that (1) characterize, up to quadratic blowups, every finite level of the OIP hierarchy in terms of other well-known communication complexity classes, (2) separate the first four levels of the hierarchy, and (3) reveal that the hierarchy collapses to the fourth level. Our study of OIPs reveals marked contrasts and some parallels with the classic Turing Machine theory of interactive proofs, establishes limits on the power of existing techniques for developing constant-round SIPs, and provides a new characterization of (non-online) Arthur–Merlin communication in terms of an online model.

Key words. streaming interactive proofs, Arthur-Merlin communication complexity, probabilistic proof systems

AMS subject classifications. 68Q05, 68Q15

1. Introduction.

The surging popularity of commercial cloud computing services, and more generally outsourced computations, has revealed compelling new applications for the study of *interactive proofs* with highly restricted *verifiers*. Consider, e.g., a retailer (verifier) who lacks the resources to locally process a massive input (say, the set of all its transactions), but can access a powerful but untrusted cloud service provider (prover), who processes the input on the retailer's behalf. The verifier must work within the confines of the restrictive *data streaming* paradigm, using only a small amount of working memory. The prover must both answer queries about the input (say, "how many pairs of blue jeans have I ever sold?"), and prove that the answer is correct. We refer to this general scenario as *verifiable data stream computation*.

It is useful to look at this computational scenario as "data stream algorithms with access to a powerful (space-unlimited) prover." As is well known, most interesting data streaming

[*]A preliminary version of this paper appeared in the 2015 Conference on Computational Complexity. This work was supported in part by the Simons Institute for the Theory of Computing, and was performed while the authors were visiting the Institute.

[†]Dartmouth College, Hanover, NH, USA. Supported in part by NSF grant CCF-1217375.

[‡]University of Warwick, Coventry, UK. Supported in part by European Research Council grant ERC-2014-CoG 647557 and a Royal Society Wolfson Research Merit Award.

[§]University of Massachusetts, Amherst, Amherst, MA, USA. Supported in part by NSF grants IIS-1251110 and CCF-1637536.

[¶]Georgetown University, Washington, DC USA. This work was performed while the author was a Research Fellow at the Simons Institute for the Theory of Computing, UC Berkeley. Supported by a Research Fellowship from the Simons Institute for the Theory of Computing.

[‖]University of Utah, Salt Lake City, UT, USA. Supported in part by NSF grant IIS-1251049 and the Simons Institute for the Theory of Computing.

problems have no nontrivial (i.e., sublinear space) algorithms unless one allows approximation. For instance, given a stream σ of tokens from the universe $[n] := \{1,2,\ldots,n\}$, which implicitly defines frequencies f_j for each $j \in [n]$, some basic questions we can ask about σ are the number of distinct tokens $F_0(\sigma)$, the kth frequency moment $F_k(\sigma) = \sum_{j=1}^{n} f_j^k$, the median of the collection of numbers in σ, and the very basic *point queries* where, given a specific $j \in [n]$ after σ has been presented, we wish to know f_j. In each case, we would like an *exact* answer, not an estimate. With the trivial exception of $F_1(\sigma)$—which is just the length of σ—not one of these questions can be answered by a (possibly randomized) streaming algorithm restricted to $o(n)$ space. However, with access to a powerful prover, things improve greatly: as shown in Chakrabarti et al. [11], point queries, median, and F_k (for integral $k > 0$) can be computed exactly by a verifier using $\tilde{O}(\sqrt{n})$ space, while receiving $\tilde{O}(\sqrt{n})$ bits of "help" from the prover. Simultaneously achieving sublinear space and help is crucial because, for massive inputs, it is infeasible for the verifier to store the entire input, and for "help" messages as large as the input to be transmitted from the prover to the verifier.

Notably, the protocol achieving this $\tilde{O}(\sqrt{n})$ cost (space plus amount of help) is non-interactive: the prover sends a single message to the verifier. Chakrabarti et al. [11] also showed that under this restriction their protocol is optimal: a cost of $\Omega(\sqrt{n})$ is required. In subsequent work, Cormode et al. [17] considered *streaming interactive proofs* (SIPs), where the verifier may engage in several rounds[1] of interaction with the prover, seeking to minimize both the space used by the verifier and the total amount of communication.

Details of the SIP model. In a k-message SIP, the verifier first processes a data stream σ in a single pass, during which the verifier computes a "summary" of the stream. The space cost of computing this summary counts against the verifier's space cost in the SIP protocol. In the above example of a retailer using the cloud to store and process transactions, the retailer's pass over a stream of transactions can happen implicitly as the transactions occur, and each transaction can be immediately uploaded to the cloud (prover) so that the cloud may also learn the stream σ.

After the stream has been processed, the verifier exchanges up to k messages with the prover, who knows σ. The total length of all messages exchanged is the "help cost". The SIP model allows the verifier to process each message sent by the prover in a single streaming pass over the message, and any memory used during the pass contributes to the space cost.[2] Following this exchange of messages, the verifier must output a value, ideally equal to $g(\sigma)$, where g is the query that the verifier wants to evaluate on σ; the verifier may also choose to output a special symbol $\perp$, indicating that he suspects the prover to be cheating. Standard completeness and soundness requirements are imposed. Namely, an "honest" prover should convince the verifier to output $g(\sigma)$, and "dishonest" provers should fail to convince the verifier to output an incorrect value (see Section 2 for details). As both the space and help costs should be small, the total cost of a SIP is defined to be the sum of the space and help costs.

Prior Results on the SIP Model and Their Relationship to This Paper. Cormode et al. [17] gave SIPs in which the prover and verifier exchange $2k-1$ messages, achieving a cost of $\tilde{O}(n^{1/(k+1)})$ for the above problems. This generalizes the earlier set of results [11], which gave 1-message SIPs. Moreover, it achieves $O(\text{polylog}\, n)$ cost with $O(\log n/\log\log n)$ rounds[3] of interaction. Klauck and Prakash [29] further studied this kind of computation and

[1]Throughout, if the prover and verifier in a protocol exchange k messages, then we say that the protocol takes $\lceil k/2 \rceil$ rounds.

[2]In most of the SIPs in this paper, both the help and space costs are polylogarithmic in the input size, and hence the verifier can explicitly store and process the prover's messages while keeping the space cost polylogarithmic.

[3]For precision, throughout the paper we quantify the amount of interaction in a SIP in terms of the number of messages exchanged, as opposed to the number of rounds (since specifying the number of messages in a protocol

generalized the lower bound for non-interactive protocols, claiming that a $(2k-1)$-message SIP must cost $\Omega(n^{1/(k+1)})$, even for very basic point queries.

However, we identify an implicit assumption in the Klauck–Prakash lower bound argument: it applies only to protocols in which the verifier's messages to the prover are independent of the input. This happened to hold in all previous SIPs, which are ultimately descended from the sum-check protocol of Lund et al. [32]. Furthermore, this assumption is harmless in the classical theory of interactive proofs where public-coin protocols can simulate private-coin ones with just a polynomial blowup in cost [21]. However, these simulation results fail subtly in the streaming setting, and we show that this failure is intrinsic by giving a number of new upper bounds.

1.1. New Results: Exponentially Improved Constant-Round SIPs. We start by showing that even two-message SIPs are exponentially more powerful than previously believed, on certain problems. For now we state our results informally, using the $\tilde{O}$-notation to suppress "lower order" factors. We give formal theorem statements later in the paper, after all definitions are in place.

RESULT 1.1 (Formalized in Theorem 3.1). *There is a two-message SIP with cost* $\tilde{O}(\log n)$ *for answering point queries on a stream over the universe* $[n]$.

The SIP that achieves this upper bound is based on an abstract protocol that we call the *polynomial evaluation protocol*. Crucially, unlike the sum-check protocols used in previous SIPs, it involves an interaction where the verifier's message to the prover depends on part of the input; specifically, it depends on the query. Note that need to exchange at least two messages is likely unavoidable in practice even if verifiability is not a concern: one message may be required for the verifier to communicate the query to the prover, with a second message required for the prover to reply.

Adding a third message allows us to answer selection queries, of which an important special case is median-finding.

RESULT 1.2 (Formalized in Theorem 3.7). *There is a three-message SIP with cost* $\tilde{O}(\log n)$ *for determining the exact median of a stream of numbers from* $[n]$.

We can in fact answer fairly complex queries with three messages and polylogarithmic cost. For instance, given a data set presented as a stream of points from a metric space, we can answer *exact* nearest neighbor queries to the data set very efficiently, even in high dimensions. This is somewhat surprising, given that even the offline version of the problem seems to exhibit a curse of dimensionality.

RESULT 1.3 (Formalized in Theorem 3.4). *For data sets consisting of points from* $[n]^d$ *under certain metrics, such as the Manhattan distance* ℓ_1^d *or the Euclidean distance* ℓ_2^d*, there is a three-message SIP with cost* $\text{poly}(d, \log n)$ *allowing exact nearest neighbor queries to the data set.*

We also give similarly efficient two-message SIPs for other well-studied query problems, such as range counting queries (Theorem 3.6), where a stream of data points is followed by a query range and the goal is to determine the number of points in the range that appeared in the stream, and pattern matching queries (even with wildcards), where a streamed text is followed by a (short) query pattern. The pattern matching SIP is highlighted in the following informal result.

RESULT 1.4 (Formalized in Theorem 3.8). *There is a 2-message SIP for pattern match-*

specifies the number of rounds, but not vice versa). However, we do refer to rounds when such precision is not required.

ing with wildcards, with space and help costs at most $O(q\log(q+m))$, where q is the length of the pattern and m is the length of the stream.

Next, we work towards a detailed understanding of the subtleties of SIPs that caused the aforementioned Klauck–Prakash lower bound [29] not to apply. Our study naturally leads into communication complexity, in particular to Arthur–Merlin communication, which we discuss next.

1.2. The Connection to Arthur–Merlin Communication. Like almost all previous lower bounds for data stream computations, prior SIP lower bounds [11, 29] use reductions from problems in communication complexity. To model the prover in a SIP, the appropriate setting is Arthur–Merlin communication, which we now introduce.

Suppose Alice holds an input $x \in \mathcal{X}$, Bob holds $y \in \mathcal{Y}$, and they wish to compute $f(x,y)$ for some Boolean function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, using random coins and settling for some constant probability of error. Say this costs $\mathrm{R}(f)$ bits of communication. Can an omniscient but untrusted Merlin, who knows (x,y), convince "Arthur" (defined as Alice and Bob together) that $f(x,y) = 1$, keeping the overall communication within $o(\mathrm{R}(f))$? For several interesting functions f the answer is "Yes" and this is the general subject of Arthur–Merlin communication complexity, first considered in seminal work by Babai, Frankl, and Simon [6].

The one-pass streaming restriction on the verifier in a SIP is modeled by requiring that Alice not receive any communication from either Bob or Merlin. Thus the Alice–Bob communication is one-way, though Bob and Merlin may interact arbitrarily. We refer to this restricted communication setting as *online Arthur–Merlin communication*. It should be clear that a k-message SIP with cost C can be simulated by an online Arthur–Merlin communication of cost C where Bob and Merlin exchange k messages (see Observation 4.1 in Section 4 for details). Thus, lower bounds on SIPs would follow from corresponding communication lower bounds in the online Arthur–Merlin setting.

At this point let us recall that the classical Turing-Machine-based theory of interactive proofs considers two different models of interaction between prover and verifier, corresponding to the complexity classes $\mathbf{IP}_{\mathsf{TM}}$,[4] where the verifier is allowed private randomness, and $\mathbf{AM}_{\mathsf{TM}}$, where he may only use public randomness. Recall the following classic results about such interactive proofs.

- **Equivalence of private and public coins.** Goldwasser and Sipser [21] proved that a k-message private coin interactive proof (à la $\mathbf{IP}_{\mathsf{TM}}$) can be simulated (with a polynomial blowup in complexity) by a $(k+2)$-message public coin one (à la $\mathbf{AM}_{\mathsf{TM}}$). Thus, in the resulting protocol, the verifier can perform his interaction with the prover before even looking at the input!
- **Round reduction.** Babai and Moran [7] proved that for $k \geq 1$, a $(k+1)$-round interactive proof can be simulated by a k-round interactive proof with a polynomial blowup in the verifier's complexity. Thus, a two-message (verifier$\to$prover$\to$verifier) interactive proof is just as powerful as any constant-round one.

Interestingly, as we shall show in this work, neither of these phenomena holds for the *online* communication complexity analogs of $\mathbf{IP}_{\mathsf{TM}}$ and $\mathbf{AM}_{\mathsf{TM}}$. (Recall that "online" means that the Alice does not receive any communication from either Bob or Merlin.) This point appears to have been missed in the Klauck–Prakash proof [29], which works in a "public coin" setting and thus applies only to a restricted class of SIPs. The new SIPs we design in this work correspond to a "private coin" setting, which allows the aforementioned exponential improvements.

[4] Throughout this paper, we use the subscript "TM" to denote a Turing-machine-based complexity class, to resolve the notation clash with the analogous communication complexity classes.

Clearly there are nuances in online Arthur–Merlin communication complexity that do not arise in classical interactive proofs. In particular, we seek a better understanding of the precise role of rounds and of private randomness in the communication setting. This is the goal of our next batch of results.

1.3. New Results: Complexity Classes for Arthur–Merlin Communication. As noted above, we can think of $\mathbf{AM}_{\mathsf{TM}}$ as a restricted interactive proof model where the verifier must interact with the prover before looking at his input. We can then define a hierarchy of analogous communication complexity models called $\mathbf{OMA}^{[\mathbf{k}]}$ (Online Merlin–Arthur), where Bob and Merlin exchange k messages without looking at his input, and then Alice communicates with Bob one-way. We defer precise definitions to Section 4. The aforementioned Klauck–Prakash lower bound essentially says the following:

PROPOSITION 1.5 (Klauck and Prakash [29]). *The* INDEX *problem—where Alice gets $x \in \{0,1\}^n$, Bob gets $j \in [n]$ and Bob must output x_j with high probability—requires $\Omega(n^{1/(k+1)})$ cost in the $\mathbf{OMA}^{[\mathbf{2k}]}$ model.*

We can also define a parallel hierarchy $\mathbf{OIP}^{[\mathbf{k}]}$ (Online Interactive Proof) of communication analogs of $\mathbf{IP}_{\mathsf{TM}}$. We now hit another subtlety. We could require the Bob–Merlin interaction to happen before the Alice$\to$Bob communication; this is how we shall define $\mathbf{OIP}^{[\mathbf{k}]}$. Alternatively, we could swap the order, so that Bob's messages to Merlin could depend on Alice's input as well; we shall call the resulting (more powerful) model $\mathbf{OIP}_{+}^{[\mathbf{k}]}$.

These communication models correspond to SIPs as follows. Every SIP designed prior to this work falls into a restricted setting where the verifier's messages are independent of the input, so it can be simulated by an $\mathbf{OMA}^{[\mathbf{k}]}$ protocol with k being the number of messages exchanges by the prover and verifier in the SIP. The SIPs we design in this work apply to "query problems" with the data set appearing before the query, and our verifier messages depend only on the query. Thus our SIPs are naturally simulable by $\mathbf{OIP}^{[\mathbf{k}]}$ protocols. Finally, a general SIP, where verifier messages can depend on the entire input stream, is simulable by an $\mathbf{OIP}_{+}^{[\mathbf{k}]}$ protocol.

Following Babai et al. [6], given a communication model $\mathbf{C}$, we define a corresponding complexity class, also denoted $\mathbf{C}$, consisting of all problems that have *polylogarithmic* cost protocols in the model $\mathbf{C}$. We now have three parallel hierarchies of communication complexity classes: $\mathbf{OMA}^{[\mathbf{k}]}$, $\mathbf{OIP}^{[\mathbf{k}]}$, and $\mathbf{OIP}_{+}^{[\mathbf{k}]}$. For our next batch of results, we prove several inclusion and separation results relating these newly defined classes to each other and to well-studied classes from earlier work in communication complexity.

RESULT 1.6 (Formalized over several theorems in Section 5). *The complexity class inclusions and separations given in Figure 1.1 hold.*

Notice that there are several two-way inclusions (i.e., equalities) amongst these communication complexity classes. It is worth noting that with one exception (namely $\mathbf{OIP}^{[\mathbf{1}]} = \mathbf{OIP}_{+}^{[\mathbf{1}]}$) none of these equalities is trivial. For instance, consider the switch from the model $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ to the model $\mathbf{OIP}^{[\mathbf{2}]}$: Bob loses the ability to send Alice a message before hearing from her, but gains access to Merlin. It is not *a priori* clear that this switch in models will result in a complexity class that is even comparable to $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$, and nontrivial simulation arguments (Theorems 5.3 and 5.6) are required to prove that $\mathbf{R}^{[\mathbf{2},\mathbf{B}]} = \mathbf{OIP}^{[\mathbf{2}]}$.

Many of our simulations incur some blowup in cost. All such blowups are at most quadratic, so polylogarithmic costs remain polylogarithmic.

The $\mathbf{OMA}$ and $\mathbf{OIP}$ hierarchies behave quite differently from the classical $\mathbf{AM}_{\mathsf{TM}}$ and $\mathbf{IP}_{\mathsf{TM}}$:

- In contrast to the Goldwasser–Sipser private-by-public-coin theorem, the class $\mathbf{OIP}^{[\mathbf{4}]}$

$$\begin{tikzcd}
\mathbf{R}^{[\mathbf{1,A}]} \arrow[d, leftrightarrow] & \mathbf{R}^{[\mathbf{2,B}]} \arrow[d, leftrightarrow] & \mathbf{MA}^{[\mathbf{2,B}]} \arrow[d, leftrightarrow] & \mathbf{AM} \arrow[d, leftrightarrow] & \mathbf{OMA}^{[\mathbf{k}]} \arrow[d, two heads] \\
\mathbf{OIP}^{[\mathbf{1}]} \arrow[r, two heads] \arrow[d, leftrightarrow] & \mathbf{OIP}^{[\mathbf{2}]} \arrow[r, two heads] \arrow[d, two heads] & \mathbf{OIP}^{[\mathbf{3}]} \arrow[r, two heads] & \mathbf{OIP}^{[\mathbf{4}]} \arrow[r, leftrightarrow] & \mathbf{OIP}^{[\mathbf{k}]} \\
\mathbf{OIP}_{+}^{[\mathbf{1}]} & \mathbf{R}^{[\mathbf{3,A}]} \arrow[r] & \mathbf{OIP}_{+}^{[\mathbf{2}]} \arrow[ur] & &
\end{tikzcd}$$

Fig. 1.1: The layout of our communication complexity zoo. An arrow from $\mathbf{C}_1$ to $\mathbf{C}_2$ indicates that $\mathbf{C}_1 \subseteq \mathbf{C}_2$. If the arrow is double-headed, then the inclusion is strict. Here $k > 4$ is an arbitrary constant. The models $\mathbf{R}^{[\mathbf{t,A}]}$ (resp. $\mathbf{R}^{[\mathbf{t,B}]}$) are standard t-message randomized communication with Alice (resp. Bob) starting. The model $\mathbf{MA}^{[\mathbf{2,B}]}$ consists of a message from Merlin followed by Bob$\rightarrow$Alice$\rightarrow$Bob communication, while **AM** is standard (see Section 5).

is strictly more powerful than $\mathbf{OMA}^{[\mathbf{k}]}$ (in fact, even $\mathbf{OIP}^{[\mathbf{2}]} \not\subseteq \mathbf{OMA}^{[\mathbf{k}]}$), for every constant k.

- In contrast to the Babai–Moran round reduction theorem, there are exactly four distinct levels (not two) in the $\mathbf{OIP}^{[\mathbf{k}]}$ hierarchy, for constant k.

In the course of proving the separation results in Figure 1.1, we obtain concrete lower bounds for explicit functions that are of interest in their own right. Let us highlight one of these.

RESULT 1.7 (Formalized in Corollary 5.9). *The set disjointness problem* DISJ*—where Alice and Bob each get a subset of* $[n]$ *and must decide whether they are disjoint—requires* $\Omega(n^{1/3})$ *cost in the* $\mathbf{OIP}^{[\mathbf{3}]}$ *model and thus does not belong to the class* $\mathbf{OIP}^{[\mathbf{3}]}$*. This lower bound is tight up to a logarithmic factor.*

This has implications for SIPs. We noted that all SIPs designed thus far (including the new ones in this work) are simulable in the weaker **OIP** models. By a standard reduction [5] from DISJ to the frequency moments problem F_k, it follows that unlike what we achieved for point queries and median queries, *based on currently known techniques*, we cannot get a polylogarithmic cost three-message SIP for F_k ($k \neq 1$).

Removing the qualifier "based on currently known techniques" above would require a similar lower bound for $\mathbf{OIP}_{+}^{[\mathbf{3}]}$. Unfortunately, at present we are unable to prove any nontrivial lower bounds on $\mathbf{OIP}_{+}^{[\mathbf{2}]}$, and doing so appears to be a difficult problem (see the paragraph on Subsequent Work in Section 1.5 for details). Indeed, this inability is what led us to study the weaker **OIP** model. Yet, because the **OIP** models are online, the separation results in Figure 1.1 still morally capture the primary way in which SIPs differ from classical interactive proofs, due to SIPs' streaming/online nature,

Finally, our result $\mathbf{AM} = \mathbf{OIP}^{[\mathbf{4}]}$ gives a novel characterization of **AM** in terms of *online* communication. This is surprising because online models, where no one talks to Alice, might be expected to be too weak to capture **AM**. Proving lower bounds on **AM** is a longstanding and notoriously hard problem in communication complexity [27, 28, 31]. We believe our new characterization of **AM** is of independent interest, and may prove useful in establishing non-trivial **AM** lower bounds.

1.4. Overview of Our Techniques.

The Key Technique Underlying Our SIPs. All of our SIPs can be described in the following framework. First, for some parameter b (which is typically logarithmic in the size of the data universe) and some finite field $\mathbb{F}$, a low-degree b-variate polynomial p is identified such that the following two properties hold. (1) There is some $\mathbf{z} \in \mathbb{F}^b$ such that $g(\sigma)$ can be derived from $p(\mathbf{z})$, where g is the query of interest and σ is the data stream. Note that the polynomial p will depend on the stream σ. Moreover, $\mathbf{z}$ may depend on the query g, and $\mathbf{z}$ will not be known to the verifier before processing the stream. (2) For any point $\mathbf{r}$ chosen by the verifier prior to processing the stream, the verifier can evaluate $p(\mathbf{r})$ in small space, with a single streaming pass over σ.[5]

Once such a polynomial p is identified, we can adapt a technique of Raz [38], who used the technique to characterize the complexity class $\mathbf{IP/rpoly}_{\mathsf{TM}}$. Specifically, the verifier chooses $\mathbf{r} \in \mathbb{F}^b$ at random, evaluates $p(\mathbf{r})$ while processing the stream, and sends the prover the unique line ℓ in $\mathbb{F}^b$ passing through $\mathbf{z}$ and $\mathbf{r}$. The prover responds with a univariate polynomial h claimed to equal p restricted to ℓ. Observe that the degree of h is at most the total degree of p, and hence the help cost of the protocol is proportional to the total degree of p.

As both $\mathbf{z}$ and $\mathbf{r}$ lie on ℓ, the polynomial h implies claims about both $p(\mathbf{z})$ and $p(\mathbf{r})$. In order to probabilistically check that h is as claimed, it turns out to be enough for the verifier to confirm that the latter of these two implied claims agrees with the actual value of $p(\mathbf{r})$ that the verifier computed from the stream. If this check passes, it is safe for the verifier to output the claimed value of $p(\mathbf{z})$ implied by h.

This key technique is formalized in our Polynomial Evaluation Protocol (Theorem 2.2 in Section 2.3). With this technique in hand, the remaining effort in obtaining our SIPs for various query classes lies in identifying low-degree polynomials p satisfying Properties (1) and (2) above.

The Key Techniques Underlying Our Communication Results. Most of the key ideas in our communication results appear in our proof of the equivalence $\mathbf{R}^{[\mathbf{2},\mathbf{B}]} = \mathbf{OIP}^{[\mathbf{2}]}$. We prove this via two simulation theorems, which establish that any $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol can be simulated by an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol, and vice versa.

Roughly speaking, to simulate an $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol $\mathcal{Q}$ of cost C by an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol of cost $O(C^2)$, we invoke the Polynomial Evaluation Protocol, applied to a low-degree C-variate polynomial p that on input $\mathbf{z} \in \{0,1\}^C$, outputs a field element in $\mathbb{F}_{2^C}$ encoding the message Alice would send to Bob in $\mathcal{Q}$ if Bob first sent Alice the message $\mathbf{z}$. The total communication cost of the resulting $\mathbf{OIP}^{[\mathbf{2}]}$ protocol is $O(C^2)$ bits.

Simulating an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol $\mathcal{Q}$ of cost C for a function f by an $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol for f of cost $O(C^2)$ is more involved. To explain the ideas, it is helpful to first discuss how to obtain the known result that any $\mathbf{OMA}^{[\mathbf{1}]}$ protocol $\mathcal{Q}$ of cost C can be simulated by an $\mathbf{R}^{[\mathbf{1},\mathbf{A}]}$ protocol $\mathcal{Q}'$ of cost $O(C^2)$ [1, 11]. First, consider the following $\mathbf{OMA}^{[\mathbf{1}]}$ protocol $\mathcal{Q}_1$ whose soundness error is less than 2^{-C}. In $\mathcal{Q}_1$, Alice first repeats her part of $\mathcal{Q}$ a total of t times (using fresh randomness each time), for some $t = O(C)$. This means that in $\mathcal{Q}_1$, the Alice $\rightarrow$ Bob communication is t times larger than it is in $\mathcal{Q}$. Upon receiving a message m_M from Merlin in $\mathcal{Q}_1$, Bob accepts only if m_M would cause him to accept in a majority of the t runs of $\mathcal{Q}$ for which Alice has executed her part.

Observe that while the Alice $\rightarrow$ Bob communication cost of $\mathcal{Q}_1$ is $O(C)$ times larger than $\mathcal{Q}$, the Merlin $\rightarrow$ Bob communication cost is the same in $\mathcal{Q}$ and $\mathcal{Q}_1$. This fact, combined with the soundness error of $\mathcal{Q}_1$ being much less than 2^{-C}, implies that Merlin can simply be "cut out" of $\mathcal{Q}_1$. Specifically, Bob can ignore Merlin, and simply try *every possible* message that Merlin might send in $\mathcal{Q}_1$, and accept if and only if any of those messages would cause him to accept in $\mathcal{Q}_1$. Since there are only at most 2^C messages that Merlin can send in $\mathcal{Q}_1$, it is not

[5] In all of our protocols, the verifier can evaluate $p(\mathbf{r})$ even without knowing the query g in advance.

hard to show that this yields a valid $\mathbf{R}^{[\mathbf{1},\mathbf{A}]}$ protocol for f.

Adapting this technique to the case that $\mathcal{Q}$ is an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol instead of an $\mathbf{OMA}^{[\mathbf{1}]}$ protocol is a rather subtle endeavor. The key issue is that when $\mathcal{Q}$ is an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol, Merlin's message to Bob can depend on Bob's message to Merlin (whereas in an $\mathbf{OMA}^{[\mathbf{1}]}$ protocol, Bob does not send any message to Merlin). This means that one cannot reduce the soundness error of $\mathcal{Q}$ in the manner of $\mathcal{Q}_1$ above, unless the Alice $\to$ Bob communication cost, the Bob $\to$ Merlin, and the Merlin $\to$ Bob communication costs *all* increase by a factor of $O(C)$ relative to $\mathcal{Q}$. This in turn prevents the "cutting of Merlin out of the protocol" by trying every possible message of Merlin.

Roughly speaking, we address this issue by adding an extra message from Bob to Alice at the start of $\mathcal{Q}_1$. This message specifies $O(C)$ values of the "private randomness" in Q, all of which would lead Bob to send the *same* message to Merlin in $\mathcal{Q}$. Bob's ability to do this crucially depends on the fact that $\mathcal{Q}$ is an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol and not an $\mathbf{OIP}_{+}^{[\mathbf{2}]}$ protocol, as this ensures that Bob's message to Merlin in $\mathcal{Q}$ does not depend on Alice's message to Bob in $\mathcal{Q}$.

Simplifying a little, the above yields a (non-online) interactive proof protocol with soundness error much less than 2^{-C}, in which the Bob $\to$ Merlin and Merlin $\to$ Bob communication cost is the same as in $\mathcal{Q}$. At this point, Merlin can be cut out of the protocol exactly as in the $\mathbf{OMA}^{[\mathbf{1}]}$ case described above. The resulting Merlin-less protocol is an $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol of cost $O(C^2)$.

1.5. Related Work.

Stream Computation. Early theoretical work on verifiable stream computation focused on non-interactive protocols, as in the *annotated data streams* model of Chakrabarti et al. [11]. In our language, that model corresponds to 1-message SIPs. Work in this model has established optimal protocols for several problems including frequency moments and frequent items [11]; linear algebraic problems such as matrix rank [29]; and graph problems like shortest *s*–*t* path [16] and counting triangles [44]. Many of these protocols have subsequently been optimized for streams whose length is much smaller than the universe size [10]. More recent protocols, such as the *Arthur–Merlin streaming protocols* of Gur and Raz [10,24] are "barely interactive" in the sense that the prover and the verifier may exchange a constant number of messages. Meanwhile, the fully general *streaming interactive proof* (SIP) model of Cormode et al. [15,17] permits "many" rounds of interaction. Cormode, Thaler, and Yi [17] showed that several general $\mathbf{IP}_{\mathsf{TM}}$ protocols can be simulated in this model. These include the powerful, general-purpose protocol of Goldwasser, Kalai, and Rothblum [20]. Given any problem in $\mathbf{NC}_{\mathsf{TM}}$, the resulting protocol requires only polylogarithmic space and communication while using polylogarithmic rounds of verifier–prover interaction. Refinements and implementations of these protocols [15,43,45] have demonstrated scalability and the practicality of this line of work.

Algebraic techniques lie at the core of almost all nontrivial protocols in the above models. Specifically, a number of 1-message SIPs are inspired by the Aaronson–Wigderson **MA** communication protocol for DISJ [2], which is in turn inspired by the classic sum-check protocol of Lund et al. [32]. The sum-check protocol is also the inspiration for the way that all previous multi-round SIPs make use of interaction. The aforementioned protocol of Goldwasser et al. [20] also builds upon the sum-check protocol.

The algorithmic results outlined in Subsection 1.1 have a rather different algebraic idea at their core. They are based on the aforementioned *polynomial evaluation protocol*, which is obtained by adapting a result of Raz [38] about $\mathbf{IP/rpoly}_{\mathsf{TM}}$ to a streaming setting; see the discussion at the start of Subsection 2.3.

Early work on interactive proofs studied space-bounded verifiers (see the survey by Condon [14]), but many protocols developed in this line of work require the verifier to store the input, and therefore do not address verifiable *stream* computation, as we do here. Goldwasser et al. [19] studied interactive proofs with verifiers in the complexity class $\mathbf{NC}^0_{\mathsf{TM}}$. Interestingly, they showed that private randomness is necessary to obtain interactive proofs with verifiers in $\mathbf{NC}^0_{\mathsf{TM}}$, unless the language in question is already in $\mathbf{NC}^0_{\mathsf{TM}}$. This is analogous to our finding that constant-round "public coin" SIPs (where the verifier's messages do not depend on the input) are exponentially weaker than general constant-round SIPs.

Computationally Sound Protocols. Protocols for verifiable stream computation have also been studied in the cryptography community [12, 37, 40]. These works only require soundness to hold against cheating provers that run in polynomial time. In exchange for this weaker security guarantee, these protocols can achieve properties that are impossible in the information-theoretic setting we consider. For example, they typically achieve *reusability*, allowing the verifier to use the same randomness to answer many queries. In contrast, our protocols only support "one-shot" queries, because they require the verifier to reveal secret randomness to the prover.

Chung et al. [12] combine the GKR protocol with fully homomorphic encryption (FHE) to give reusable, non-interactive protocols of polylogarithmic cost for any problem in **NC**. Papamanthou et al. [37] give improved protocols for a class of low-complexity queries including point queries and range search: their protocols avoid the use of FHE, and allow the prover to answer such queries in polylogarithmic time (a similar property was achieved by Schröder and Schröder [40], but for a simpler class of queries, and with unidirectional communication from the verifier to the prover on each stream update). In contrast, prior work as well as our own requires the prover to spend time quasilinear in the size of the data stream after receiving a query, even if the answer itself can be computed in sublinear time (e.g., point queries, which can be solved with a single access to memory). We note however that our most interesting protocols, such as those for nearest neighbor search and pattern matching, are for problems that cannot be solved in sublinear time; hence, the quasilinear time required by our protocols does not affect the prover's runtime by more than logarithmic factors.

Communication Complexity. Seminal work by Babai et al. [6] introduced and studied the communication analogs of the major Turing Machine complexity classes, including **P**, **NP**, $\mathbf{\Sigma}_2$, $\mathbf{\Pi}_2$. They hinted at similar analogs of **MA** and the **AM** hierarchy. Lokam [31] related the task of placing problems outside of the communication class **AM** to notions of matrix rigidity. He also observed that the communication complexity classes **IP** and **AM** behave similarly to their Turing Machine counterparts. In particular, noted theorems such as $\mathbf{IP} = \mathbf{PSPACE}$, Toda's Theorem, and Babai and Moran's round reduction results [7] all hold in the communication world (though not under *online* communication, as shown by this work).

Online (also known as one-message) randomized communication complexity was introduced in the mid-1990s and considered by Ablayev [4], Kremer, Nisan, and Ron [30], and Newman and Szegedy [34]. Aaronson [1] introduced online variants of Merlin–Arthur communication, in classical and quantum flavors. Aaronson and Wigderson [2] gave an online **MA** communication protocol for DISJ (more generally, for INNER-PRODUCT) with cost $\tilde{O}(\sqrt{n})$; this is nearly optimal, as shown by a lower bound of Klauck [27] that applies to general **MA** protocols. More recently, Klauck [28] performed a careful study of **AM**, **MA**, and its quantum analogue **QMA**. In particular, he gave a promise problem PAPPMP separating **QMA** from **AM**; we shall eventually show that PAPPMP separates $\mathbf{OIP}^{[3]}$ from $\mathbf{OIP}^{[4]}$.

Subsequent Work. Subsequent to the conference version of this paper, Daruki et al. [18] gave SIPs for a number of problems in computational geometry and data analysis. In two of their results, they built on our work to give a 2-message SIP of polylogarithmic cost for the Minimum Enclosing Ball problem, and a 3-message SIP of polylogarithmic cost for computing a 2-approximation to the k-center in a metric space. Abdullah et al. [3] gave logarithmic round SIPs of polylogarithmic cost (plus the cost of specifying an optimal solution) for a number of graph problems.

Very recently, Bouland et al. [9] have "explained" our inability to prove lower bounds on $\mathbf{OIP}_{+}^{[2]}$ protocols: they showed that the $\mathbf{OIP}_{+}^{[2]}$ model, as well as 2-message SIPs themselves, are powerful enough to compute (partial) functions outside of **UPP**. **UPP** is the most powerful two-party communication model against which existing methods can prove lower bounds (see, e.g., [22]). Hence, proving $\mathbf{OIP}_{+}^{[2]}$ lower bounds is likely to require substantially new techniques.

1.6. Suggestions for Reading the Rest of the Paper. As should be clear by now, this paper contains two groups of results. The first group provides upper bounds by designing SIPs. The reader primarily interested in this group can simply continue with Sections 2 and 3. The second group concerns Arthur–Merlin communication, lower bounds, and a number of structural complexity results. The reader primarily interested in these results should first study the polynomial evaluation protocol, discussed in Section 2, and may then skip to Sections 4 and 5. This order is important: several of the complexity results make use of the polynomial evaluation protocol.

Section 5 contains a large number of individual theorems. To the reader wishing to get a small but representative sampling of the salient techniques, we suggest Theorems 5.3 and 5.6 and Proposition 5.13.

2. The SIP Model and the Polynomial Evaluation Protocol.

2.1. The SIP Model. In a data stream problem, the input σ is a *stream*, or sequence, of *tokens* from some data universe $\mathcal{U}$. The goal is to compute or approximate some function $g(\sigma)$, keeping space usage sublinear in the two key size parameters: (1) the length of σ, and (2) the size of the universe $|\mathcal{U}|$. Practically speaking, we would also like to process each stream update (token arrival) quickly. All our data stream algorithms will be randomized, and we shall allow them to err with some small constant probability on each input stream. In the *streaming interactive proofs* (SIP) model, after processing σ, the algorithm (called the "verifier") may exchange k messages with an entity (the "prover") who knows σ and whose goal is to lead the verifier to output the correct answer $g(\sigma)$. The SIP model allows the verifier to process each message sent by the prover in a single streaming pass over the message; any memory used during the pass contributes to the space cost. The verifier, being distrustful, will output "$\perp$" (indicating "abort") if he suspects the prover to be cheating.

All of the SIPs in this paper will work in the *turnstile streaming model*, where σ can contain deletions of tokens from $\mathcal{U}$, in addition to insertions. In this model it is best to think of the input as being a stream of integer *updates* to a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$. Initially $\mathbf{x} = \mathbf{0}$, and an update is a tuple $(i, c) \in [n] \times \mathbb{Z}$, which has the effect of adding c to the entry x_i. We will sometimes describe our algorithms as they apply to the vanilla streaming model, but it will be straightforward to extend them to the turnstile model.

We say that a SIP computes the function g with *completeness error* ε_c and *soundness error* ε_s if for all inputs $\mathbf{x}$ there exists a prover strategy that will cause the verifier to output $g(\mathbf{x})$ with probability at least $1 - \varepsilon_c$, and no prover strategy can cause the verifier to output a value outside $\{g(x), \perp\}$ with probability larger than ε_s. In designing SIPs, our goal will be to achieve $\varepsilon_c, \varepsilon_s \leq 1/3$; clearly the theory remains unchanged if we replace $1/3$ by another constant in $(0, 1/2)$. A SIP with $\varepsilon_c = 0$ is said to have *perfect completeness*. The total length of the verifier–prover interaction is the *help cost*. The space used by the streaming verifier to process both the input and the prover's messages is the *space cost*. The *cost* of a SIP is the sum of its help cost and its space cost. When designing SIP protocols we will also discuss the time complexities of the prover and the verifier. To keep things simple, we consider a model in which all arithmetic operations on a finite field of size $n^{O(1)}$ can be executed in unit time.

2.2. Low-Degree and Multilinear Extensions. A recurring technical element in our SIPs is the notion of the *low-degree extensions* of any function defined over the Boolean hypercube. Let $\mathbb{F}$ be any field, and let $f\colon \{0,1\}^b \to \mathbb{F}$ be any function. A b-variate polynomial $\widetilde{f}$ over $\mathbb{F}$ is said to *extend* f if for all $\mathbf{u} \in \{0,1\}^b$, $\widetilde{f}(\mathbf{u}) = f(\mathbf{u})$. A low-degree extension of f is any extension that has "low" degree in each variable. Generally speaking, "low-degree" in this work will mean $\mathrm{poly}(b)$.

For any function $f\colon \{0,1\}^b \to \mathbb{F}$, there is a particular low-degree extension of f that will play an especially prominent role in this work. Specifically, any f has a unique *multilinear* polynomial that extends it.[6] For obvious reasons, this polynomial is referred to as the *multilinear extension* of f. The multilinear extension of f can be expressed as follows:

$$\widetilde{f}(Z_1,\ldots,Z_b) = \sum_{\mathbf{z}\in\{0,1\}^b} f(\mathbf{z})\chi_{\mathbf{z}}(Z_1,\ldots,Z_b)\,, \text{ where} \tag{2.1}$$

$$\chi_{\mathbf{u}}(Z_1,\ldots,Z_b) = \prod_{i=1}^{b}\big((1-u_i)(1-Z_i)+u_iZ_i\big)\,. \tag{2.2}$$

It is straightforward to check that the polynomial $\widetilde{f}$ defined in (2.1) is multilinear, and that it extends f.

2.3. The Polynomial Evaluation Protocol. We shall present a two-message SIP for an abstract data stream problem called "polynomial evaluation," where the input consists of a multivariate polynomial described implicitly, as a table of values, followed by a point at which the polynomial must be evaluated. Without space constraints, this problem simply amounts to interpolation followed by direct evaluation, but our goal is to obtain a protocol where the verifier uses space roughly logarithmic in the size of the table of values, and is convinced by the prover about the correct answer after a similar amount of communication. For ease of presentation, we shall first consider a concrete special setting that is important in its own right: the INDEX problem. Recall that in this problem, the input is a stream of *n data bits* $x_1,\ldots,x_n$, followed by a *query index* $j \in [n]$. The goal is to output x_j with error at most $1/3$.

With very different motivations from ours, Raz [38] gave an interactive proof protocol placing *every* language in $\mathbf{IP}_{\mathsf{TM}}/\mathbf{rpoly}$, the class of languages that have interactive proofs with polynomial-time verifiers that take randomized advice, where the advice is kept secret from the prover. Our SIP for INDEX can be seen as an adaptation of Raz's interactive proof to the streaming setting.

THEOREM 2.1. *The* INDEX *problem has a two-message SIP with cost $O(\log n \log\log n)$, in which the verifier processes each stream token in $O(\log n)$ time and the prover runs in total time $O(n\log n)$.*

Proof. Assume WLOG that $n = 2^b$, for some integer b. Identify each integer $z \in [n]$ with a Boolean vector $\mathbf{z} = (z_1,\ldots,z_b) \in \{0,1\}^b$ in some canonical way, such as by using the binary representation of z. We can then view the data bits as a table of values for the Boolean function $g_x : \{0,1\}^b \to \{0,1\}$ given by $g_x(\mathbf{z}) = x_z$. We shall let $\mathbb{F}[Z_1,\ldots,Z_b]$ be a fixed "large enough" finite field $\mathbb{F}$, and let $\widetilde{g}_x$ denote the multilinear extension of g_x over $\mathbb{F}$. We define a *line* in $\mathbb{F}^b$ to be the range of a nonconstant affine function from $\mathbb{F}$ to $\mathbb{F}^b$. Every line contains exactly $|\mathbb{F}|$ points. Given such a line, ℓ, we define its *canonical representation* to be the degree-1 polynomial $\lambda_\ell(W) \in \mathbb{F}^b[W]$ such that $\lambda_\ell(0)$ and $\lambda_\ell(1)$ are, respectively, the lexicographically

[6] A multilinear polynomial is a polynomial with degree at most 1 in each variable.

first and second points in ℓ. We define the *canonical restriction* of a polynomial $f(Z_1,\ldots,Z_b)$ to ℓ to be the univariate polynomial $f(\lambda_\ell(W)) \in \mathbb{F}[W]$, whose degree is at most the total degree of f.

Using the above notations and conventions, our two-message SIP for INDEX works as shown in Figure 2.1.

Input: Stream of data bits $(x_1,\ldots,x_n)$ where $n = 2^b$, followed by index $j \in [n]$.
Goal: Prover to convince Verifier to output the correct value of x_j.
Shared Agreement: Finite field $\mathbb{F}$ with $3b+1 \le |\mathbb{F}| \le 6b+2$; bijective map $u \in [n] \longleftrightarrow \mathbf{u} \in \{0,1\}^b$.

Initialization: Verifier picks $\mathbf{r} \in_R \mathbb{F}^b$ uniformly at random, sets $Q \leftarrow 0$.
Stream Processing: Upon reading x_z, where $z \in [n]$, Verifier updates $Q \leftarrow Q + x_z\chi_{\mathbf{z}}(\mathbf{r})$.
Query Handling: Upon reading the index j, Verifier interacts with Prover as follows:

1. If $\mathbf{j} = \mathbf{r}$, Verifier outputs Q as the answer. Otherwise, he sends Prover ℓ, the unique line in $\mathbb{F}^b$ through $\mathbf{j}$ and $\mathbf{r}$.
2. Prover sends Verifier a polynomial $h(W) \in \mathbb{F}[W]$ of degree at most b, claiming that it is the canonical restriction of the multilinear polynomial $\widetilde{g}_x(Z_1,\ldots,Z_b)$ to the line ℓ. That is, Prover claims that $h(W) \equiv \widetilde{g}_x(\lambda_\ell(W))$.
3. Let $w,t \in \mathbb{F}$ be such that $\lambda_\ell(w) = \mathbf{j}$ and $\lambda_\ell(t) = \mathbf{r}$. Verifier checks that $h(t) = Q$, aborting if not. If the check passes, Verifier outputs $h(w)$ as the answer.

Fig. 2.1: A Two-Message Streaming Interactive Proof (SIP) Protocol for the INDEX Problem

To analyze this protocol, first note that after reading all the data bits, the verifier would have computed $Q = \widetilde{g}_x(\mathbf{r})$, by (2.1). Now the protocol is easily seen to have perfect completeness. Since $\widetilde{g}_x(Z_1,\ldots,Z_b)$ is multilinear, it follows that $\deg(\widetilde{g}_x(\lambda_\ell(W))) \le b$, so the prover can always honestly choose $h(W) = \widetilde{g}_x(\lambda_\ell(W))$. If he does so, then we will indeed have $h(t) = \widetilde{g}_x(\lambda_\ell(t)) = \widetilde{g}_x(\mathbf{r}) = Q$, and the verifier's check will pass. Finally, the verifier will output $h(w) = \widetilde{g}_x(\lambda_\ell(w)) = \widetilde{g}_x(\mathbf{j}) = x_j$, the correct answer to the INDEX instance.

Next, we analyze soundness. If the prover supplies a polynomial $h(W) \not\equiv \widetilde{g}_x(\lambda_\ell(W))$, then, since both polynomials have degree at most b, they agree at no more than b points in $\mathbb{F}$. From the prover's perspective after he receives the verifier's message, $\mathbf{r}$ is uniformly distributed in $\ell \setminus \{\mathbf{j}\}$. Thus, $\Pr_{\mathbf{r}}[h(t) = Q] \le b/(|\mathbb{F}|-1) \le 1/3$.

Now we consider this protocol's costs. The verifier maintains the random point $\mathbf{r} \in \mathbb{F}^b$ and the running sum $Q \in \mathbb{F}$, using $O(b\log|\mathbb{F}|)$ space. He sends the prover ℓ, which is specified by two elements of $\mathbb{F}^b$, and receives a degree-b polynomial in $\mathbb{F}[W]$; both communications use at most $O(b\log|\mathbb{F}|)$ bits. Recalling that $|\mathbb{F}| \le 6b+2$, we see that both space and communication costs are in $O(b\log b) = O(\log n \log\log n)$.

Finally, we consider the verifier's and prover's runtimes. The honest prover must send the univariate polynomial $\widetilde{g}_x(\lambda_\ell(W))$. Since $\widetilde{g}_x$ has degree at most b, it suffices for the prover to specify the evaluations of $\widetilde{g}_x(\lambda_\ell(W))$ at $b+1 = O(\log n)$ points. A direct application of (2.1) and (2.2) shows that each evaluation can be done in $O(n\log n)$ time, resulting in a total runtime of $O(n\log^2 n)$. However, using now-standard memoization techniques (see, e.g., [45, Section 5.1]), it is possible for the prover to in fact perform each of these evaluations in just $O(n)$ time, resulting in a total runtime of $O(n\log n)$. The verifier can run in $O(b) = O(\log n)$ time per stream update, as each stream update x_z only requires the verifier to compute $\chi_{\mathbf{z}}(\mathbf{r})$, and it follows from (2.2) that this can be done with $O(b)$ field operations. When interacting with the prover, the verifier first needs to determine the line ℓ through $\mathbf{j}$ and $\mathbf{r}$, which he can do in

$O(b) = O(\log n)$ time. To process the prover's reply, he must evaluate the polynomial h at the points t and w; these evaluations can be done in $\operatorname{polylog} n$ time. ◻

The above SIP protocol uses very little of the special structure of the INDEX problem. Let us abstract out its salient features, so as to handle the general problem described at the start of this section. First, note the protocol treats the data set given by $(x_1, \ldots, x_n)$ as an implicit description of the polynomial $\widetilde{g}_x$. Second, note that our soundness analysis did not require multilinearity per se, only an upper bound on the total degree of $\widetilde{g}_x$. Finally, note that the specific form of (2.1) and (2.2) is not crucial either; all we used was that it allows the verifier an easy streaming computation. Thus, we obtain the following generic result.

THEOREM 2.2 (Polynomial Evaluation Protocol). *Suppose an input data stream implicitly describes a v-variate polynomial g of total degree d over a field $\mathbb{F}$, followed by a point $\mathbf{j} \in \mathbb{F}^v$. Suppose this implicit description allows a streaming verifier to evaluate g at a random point $\mathbf{r} \in_R \mathbb{F}^v$ using space S. Then the technique of the protocol in Figure 2.1 gives a two-message SIP for computing $g(\mathbf{j})$, with the following properties: (1) perfect completeness; (2) soundness error bounded by $d/(|\mathbb{F}|-1)$; (3) space usage in $O(v \log |\mathbb{F}| + S)$; (4) help cost in $O((d+v)\log|\mathbb{F}|)$.*

We shall refer to the abstract protocol given by Theorem 2.2 as the polynomial evaluation protocol.

Discussion: On Using Multilinear Vs. General Low-Degree Extensions. The help cost of the Polynomial Evaluation Protocol grows linearly with the degree of the polynomial g to which it is applied. Hence, to control this cost, we seek to minimize the degree of the polynomial g to which we apply the protocol. In most of our SIPs, the polynomial g to which we apply the Polynomial Evaluation Protocol is derived from a low-degree extension of some function $f\colon \{0,1\}^b \to \mathbb{F}$. The *multilinear* extension of f is the lowest degree extension of f (it has degree at most 1 in each variable). For this reason, whenever possible, we use multilinear rather than general low-degree extensions within our SIPs. However, for some of our protocols (e.g., Theorem 3.4, Theorem 3.6, and Theorem 3.7), we will be forced to use higher degree extensions if we want to ensure that the verifier runs quickly. This increases the help cost of the protocols, and it will be necessary to carefully control this increase. This will typically entail identifying an extension of f that (1) has reasonably low degree and (2) the verifier can evaluate quickly.

3. Constant-Round SIPs for Query Problems.

We shall now apply the polynomial evaluation protocol to design SIPs proving the various upper bounds outlined in Subsection 1.1. The first application is immediate; later applications bring in additional ideas.

3.1. Point Queries.

In the POINTQUERY problem, the input is a stream in the turnstile model, updating an initially-zero vector $\mathbf{x} \in \mathbb{Z}^n$, followed by a query $j \in [n]$. The goal is to output x_j.

THEOREM 3.1. *Suppose the input to* POINTQUERY *is guaranteed to satisfy $|x_i| \le q$ at the end of the data stream, for all entries of $\mathbf{x}$, where the bound q is known a priori. Then there is a two-message SIP for* POINTQUERY *with space and help costs in $O(\log n \log(q + \log n))$.*

Proof. Assume WLOG that $n = 2^b$ for an integer b, and use a bijection $u \in [n] \longleftrightarrow \mathbf{u} \in \{0,1\}^b$ as in Theorem 2.1. The vector $\mathbf{x}$ resulting from the updates defines a multilinear polynomial $\widetilde{g}_{\mathbf{x}}(Z_1, \ldots, Z_b)$ by (2.1), where $g_{\mathbf{x}}(\mathbf{z}) := x_z$. We can treat $\widetilde{g}_{\mathbf{x}}$ as a polynomial over any field we like, but to solve our problem, we need to tell apart the $2q+1$ possible values taken on by the entries of $\mathbf{x}$ (recall that q is an upper bound on $\|\mathbf{x}\|_\infty$ at the end of the stream). For this it suffices to have $\operatorname{char}(\mathbb{F}) \ge 2q+1$.

Applying the polynomial evaluation protocol is now straightforward. The verifier starts with $\mathbf{r} \in_R \mathbb{F}^b$ and $Q = 0$. Upon receiving an update indicating "$x_i \leftarrow x_i + c$," he updates $Q \leftarrow Q + c\chi_{\mathbf{i}}(\mathbf{r})$. The other details are as in Figure 2.1. The space and communication costs are both in $O(b\log|\mathbb{F}|)$ as before.

To ensure a soundness error of at most $1/3$, we let $|\mathbb{F}| > 3b$ as before. This and the earlier condition on $\mathrm{char}(\mathbb{F})$ can both be satisfied by, e.g., taking $\mathbb{F} = \mathbb{F}_p$, for a prime $p > 3b + 2q$. This translates to cost bounds in $O(\log n \log(q + \log n))$, as claimed. $\square$

3.2. Nearest Neighbor Queries. Consider a "premetric" space[7] $(\mathcal{X}, D)$ given by a finite ground set $\mathcal{X}$ and distance function $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ satisfying $D(\mathbf{x},\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$. Let $B_D(\mathbf{z}, r) = \{\mathbf{x} \in \mathcal{X} : D(\mathbf{x},\mathbf{z}) \le r\}$ denote the corresponding ball of radius $r \in \mathbb{R}^+$ centered at $\mathbf{z} \in \mathcal{X}$. In the NEARESTNEIGHBOR problem, the input consists of a stream $\langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \rangle$ of m points from $\mathcal{X}$, constituting the data set, followed by a query point $\mathbf{z} \in \mathcal{X}$. The goal is to output $\mathbf{x}^\star = \arg\min_{\mathbf{x}^{(i)}} D(\mathbf{x}^{(i)}, \mathbf{z})$, the nearest neighbor of $\mathbf{z}$ in the data set. We shall give highly efficient SIPs for this problem that handle rather general distance functions D. To keep our statements of bounds simple, we shall impose the following structure on $(\mathcal{X}, D)$.

- We assume that $\mathcal{X} = [n]^d$. We think of d as the dimensionality of the data, and $[n]^d$ as a very fine "grid" over the ambient space of possible points.
- For all $\mathbf{x},\mathbf{y} \in [n]^d$, $D(\mathbf{x},\mathbf{y}) \le 1$ is an integer multiple of a small parameter $\varepsilon \ge 1/n^d$.

Overall, this amounts to assuming that our data set has polynomial *spread*: the ratio between the maximum and minimum distance. We proceed to give two SIPs for NEARESTNEIGHBOR. Our basic SIP has cost roughly logarithmic in the stream length and the spread (and therefore linear in d but only logarithmic in n). After we present it, we shall critique it and then give a more sophisticated SIP to handle its faults.

THEOREM 3.2. *Under the above assumptions on the premetric space $(\mathcal{X}, D)$, the* NEARESTNEIGHBOR *problem has a three-message SIP with cost $O(d\log n \log(m + \log(d\log n)))$.*

Proof. Let $\mathcal{B} = \{B_D(\mathbf{x}, j\varepsilon) : \mathbf{x} \in \mathcal{X}, j \in \mathbb{Z}, 0 \le j \le 1/\varepsilon\}$ be the set of all balls of all radii between 0 and 1 (quantized at granularity ε). By our assumptions on the structure of $(\mathcal{X}, D)$, we have $|\mathcal{B}| \le n^d/\varepsilon \le n^{2d}$. The input stream $\langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \rangle$ defines a *derived stream*, consisting of updates to a vector $\mathbf{v}$ indexed by the elements of $\mathcal{B}$. We shall denote by $v[\boldsymbol{\beta}]$ the entry of $\mathbf{v}$ indexed by $\boldsymbol{\beta} \in \mathcal{B}$. The derived stream is defined as follows: the token $\mathbf{x}^{(i)}$ increments $v[\boldsymbol{\beta}]$ for every ball $\boldsymbol{\beta}$ that contains $\mathbf{x}^{(i)}$. The verifier runs the POINTQUERY protocol of Theorem 3.1 on this derived stream.

The verifier learns the query point $\mathbf{z}$ at the end of the stream. The prover then supplies a point $\mathbf{y}$ claimed to be a valid nearest neighbor (note that there may be more than one valid answer). To check this claim, it is sufficient for the verifier to check two properties: (1) that $\mathbf{y}$ did appear in the stream, and (2) that the stream contained no point closer to $\mathbf{z}$ than $\mathbf{y}$. The first property holds iff $v[B_D(\mathbf{y}, 0)] \ne 0$. The second property holds iff $v[B_D(\mathbf{z}, D(\mathbf{y},\mathbf{z}) - \varepsilon)] = 0$. Clearly, these two properties can be checked by two point queries over the derived stream.

Following the protocol of Theorem 3.1, the two point queries (executed in parallel) involve the exchange of two more messages between the verifier and the prover, for an overall three-message SIP. Since the entries of $\mathbf{v}$ never exceed m, each POINTQUERY protocol requires space and help costs $O(d\log n \log(m + \log(d\log n)))$. $\square$

While the protocol of Theorem 3.2 achieves very small space and help costs, the prover's and verifier's runtimes could be as high as $\Omega(n^d)$, because processing a single stream token $\mathbf{x}^{(i)}$ may require both parties to enumerate all balls containing $\mathbf{x}^{(i)}$. Ultimately, this inefficiency is because the protocol assumes hardly anything about the nature of the distance

[7] This very general setting, which includes metric spaces as special cases, captures several important distance functions such as the Bregman divergences from information theory and machine learning that satisfy neither symmetry nor the triangle inequality.

function D and, as a result, does not get to exploit any structural information about the balls in $\mathcal{B}$.

To rectify this, we shall make the entirely reasonable assumption that the distance function D is "efficiently computable" in the rather mild sense that membership in a ball generated by D can be decided by a short (say, polynomial-length) formula. Accordingly, we shall express our bounds in terms of a parameter that captures this notion of efficient computation.

DEFINITION 3.3. *Suppose the distance function D on $\mathcal{X}$ satisfies the assumptions for Theorem 3.2. Let $\Phi_D : \mathcal{B} \times \mathcal{X} \to \{0,1\}$ be the ball membership function for D, i.e., $\Phi_D(B_D(\mathbf{z},r),\mathbf{x}) = 1 \iff \mathbf{x} \in B_D(\mathbf{z},r)$. Think of Φ_D as a Boolean function of $(3d\log n)$-bit inputs. We define the* formula size complexity *of D, denoted $\operatorname{fsize}(D)$, to be the length of the shortest de Morgan formula for Φ_D.*

Since addition and multiplication of b-bit integers can both be computed by Boolean circuits in depth $\log b$ (see, e.g., [36, 46]), they can be computed by Boolean formulae of size $\operatorname{poly}(b)$. It follows that for many natural distance functions D, including the Euclidean, Hamming, ℓ_1, and ℓ_∞ metrics (and in fact ℓ_p for all suitably "small" positive p), we have $\operatorname{fsize}(D) = \operatorname{poly}(d, \log n)$.

THEOREM 3.4. *Suppose the premetric space $(\mathcal{X}, D)$ satisfies the assumptions made for Theorem 3.2. Then* NEARESTNEIGHBOR *on $(\mathcal{X}, D)$ has a three-message SIP, whose space and help costs are both at most $O(\operatorname{fsize}(D)\log(m+\operatorname{fsize}(D)))$, in which the verifier processes each stream update in time $O(\operatorname{fsize}(D))$, and the prover runs in total time $m \cdot \operatorname{poly}(\operatorname{fsize}(D))$. In particular, if $\operatorname{fsize}(D) = \operatorname{poly}(d, \log n)$, as is the case for many natural distance functions D, then the space and help costs are both $\operatorname{poly}(d, \log m, \log n)$, the verifier runs in time $\operatorname{poly}(d, \log n)$ per stream update, and the prover runs in total time $m \cdot \operatorname{poly}(d, \log n)$.*

Before describing the protocol in detail, let us explain the high level idea that allows us to avoid the high runtimes of the previous protocol. Essentially, the SIP of Theorem 3.2 ran our polynomial evaluation protocol on a *multilinear* extension of the vector $\mathbf{v}$ defined by the derived stream. That SIP took $\mathbf{v}$ to be a completely arbitrary table of values. As a result, the verifier's computation—evaluating the multilinear extension at a random point—became costly. The honest prover incurred similar costs. A closer examination of the nature of $\mathbf{v}$ reveals that if D is a "reasonable" distance function, then $\mathbf{v}$ itself has plenty of structure. In particular, an appropriate *higher degree* extension of $\mathbf{v}$ can in fact be evaluated much more efficiently (by both the verifier and the prover) than the above multilinear extension. Details follow.

Proof. Put $b = d\log n$ and $S = \operatorname{fsize}(D)$. According to Definition 3.3, the function Φ_D is computed by a length-S formula that takes a $2b$-bit input $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{2b})$ describing a ball in $\mathcal{B}$ and a b-bit input $\mathbf{x} = (x_1, \ldots, x_b)$ describing a point in $\mathcal{X}$. With each gate G of this formula we associate a polynomial $\widetilde{G}$ in the variables $W_1, \ldots, W_{2b}, X_1, \ldots, X_b$, as follows:

$$
\begin{aligned}
G = \beta_i &\implies \widetilde{G} = W_i\,,\\
G = x_i &\implies \widetilde{G} = X_i\,,\\
G = \neg G_1 &\implies \widetilde{G} = -\widetilde{G}_1\,,\\
G = G_1 \wedge G_2 &\implies \widetilde{G} = \widetilde{G}_1\widetilde{G}_2\,,\\
G = G_1 \vee G_2 &\implies \widetilde{G} = 1-(1-\widetilde{G}_1)(1-\widetilde{G}_2)\,.
\end{aligned}
$$

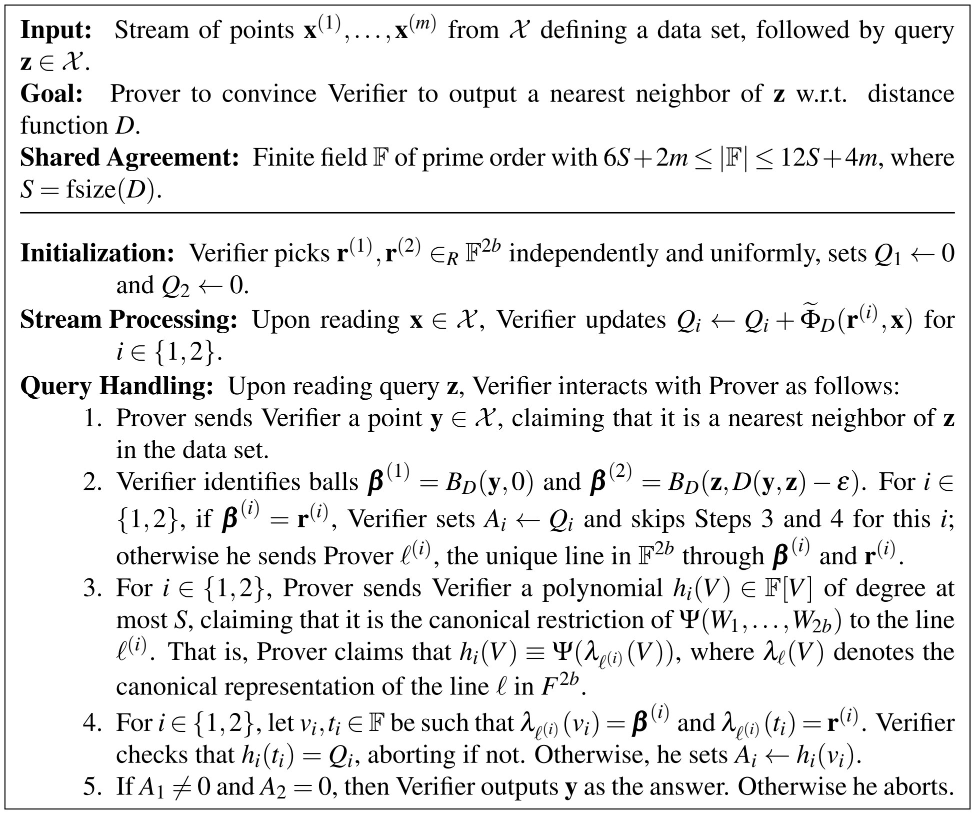
Let $\widetilde{\Phi}_D(W_1, \ldots, W_{2b}, X_1, \ldots, X_b)$ denote the polynomial thereby associated with the output gate; this polynomial is the standard arithmetization [41] of the formula. We will interpret $\widetilde{\Phi}_D$

as a polynomial in $\mathbb{F}[W_1,\ldots,X_1,\ldots]$ for a "large enough" finite field $\mathbb{F}$. By construction, $\widetilde{\Phi}_D$ has total degree at most S and agrees with Φ_D on every Boolean input. Define the polynomial $\Psi(W_1,\ldots,W_{2b}) = \sum_{i=1}^m \widetilde{\Phi}_D(W_1,\ldots,W_{2b},\mathbf{x}^{(i)})$.

Observe that the vector $\mathbf{v}$ defined by the derived stream in the proof of Theorem 3.2 behaves as follows:

$$v[\boldsymbol{\beta}] = \sum_{i=1}^m \Phi_D(\boldsymbol{\beta},\mathbf{x}^{(i)}) = \sum_{i=1}^m \widetilde{\Phi}_D(\boldsymbol{\beta},\mathbf{x}^{(i)}) = \Psi(\boldsymbol{\beta})\,. \tag{3.1}$$

Thus, Ψ is a degree-S extension of $\mathbf{v}$ to $\mathbb{F}$. The input stream defines Ψ implicitly, and the verifier can easily evaluate $\Psi(\mathbf{r})$ for random $\mathbf{r} \in_R \mathbb{F}^{2b}$. So we can invoke the polynomial evaluation protocol (twice, in parallel) and answer the NEARESTNEIGHBOR query just as in Theorem 3.2. For full clarity, we spell out the resulting SIP below. The term "canonical representation" and notation λ_ℓ are as in Subsection 2.3 and Figure 2.1.


Input: Stream of points $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)}$ from $\mathcal{X}$ defining a data set, followed by query $\mathbf{z} \in \mathcal{X}$.
Goal: Prover to convince Verifier to output a nearest neighbor of $\mathbf{z}$ w.r.t. distance function D.
Shared Agreement: Finite field $\mathbb{F}$ of prime order with $6S+2m \le |\mathbb{F}| \le 12S+4m$, where $S = \mathrm{fsize}(D)$.

Initialization: Verifier picks $\mathbf{r}^{(1)},\mathbf{r}^{(2)} \in_R \mathbb{F}^{2b}$ independently and uniformly, sets $Q_1 \leftarrow 0$ and $Q_2 \leftarrow 0$.
Stream Processing: Upon reading $\mathbf{x} \in \mathcal{X}$, Verifier updates $Q_i \leftarrow Q_i + \widetilde{\Phi}_D(\mathbf{r}^{(i)},\mathbf{x})$ for $i \in \{1,2\}$.
Query Handling: Upon reading query $\mathbf{z}$, Verifier interacts with Prover as follows:

1. Prover sends Verifier a point $\mathbf{y} \in \mathcal{X}$, claiming that it is a nearest neighbor of $\mathbf{z}$ in the data set.
2. Verifier identifies balls $\boldsymbol{\beta}^{(1)} = B_D(\mathbf{y},0)$ and $\boldsymbol{\beta}^{(2)} = B_D(\mathbf{z},D(\mathbf{y},\mathbf{z})-\varepsilon)$. For $i \in \{1,2\}$, if $\boldsymbol{\beta}^{(i)} = \mathbf{r}^{(i)}$, Verifier sets $A_i \leftarrow Q_i$ and skips Steps 3 and 4 for this i; otherwise he sends Prover $\ell^{(i)}$, the unique line in $\mathbb{F}^{2b}$ through $\boldsymbol{\beta}^{(i)}$ and $\mathbf{r}^{(i)}$.
3. For $i \in \{1,2\}$, Prover sends Verifier a polynomial $h_i(V) \in \mathbb{F}[V]$ of degree at most S, claiming that it is the canonical restriction of $\Psi(W_1,\ldots,W_{2b})$ to the line $\ell^{(i)}$. That is, Prover claims that $h_i(V) \equiv \Psi(\lambda_{\ell^{(i)}}(V))$, where $\lambda_\ell(V)$ denotes the canonical representation of the line ℓ in F^{2b}.
4. For $i \in \{1,2\}$, let $v_i,t_i \in \mathbb{F}$ be such that $\lambda_{\ell^{(i)}}(v_i) = \boldsymbol{\beta}^{(i)}$ and $\lambda_{\ell^{(i)}}(t_i) = \mathbf{r}^{(i)}$. Verifier checks that $h_i(t_i) = Q_i$, aborting if not. Otherwise, he sets $A_i \leftarrow h_i(v_i)$.
5. If $A_1 \neq 0$ and $A_2 = 0$, then Verifier outputs $\mathbf{y}$ as the answer. Otherwise he aborts.

Fig. 3.1: A Three-Message SIP for the NEARESTNEIGHBOR Problem

This protocol's correctness can be analyzed using the same ideas as in the proofs of Theorems 2.1 and 3.2. It has perfect completeness. If a dishonest prover supplies an incorrect polynomial for either $h_1(V)$ or $h_2(V)$, the verifier will fail to notice with probability at most $S/(|\mathbb{F}|-1) \le 1/6$, leading to a soundness error of at most $1/6+1/6 = 1/3$. Entries of $\mathbf{v}$ always lie between 0 and m and $\mathrm{char}(\mathbb{F}) = |\mathbb{F}| > m$, since $\mathbb{F}$ is of prime order, therefore there are no "wrap around" problems in (3.1).

Turning to the protocol's costs, the verifier needs $O(S)$ space to evaluate $\widetilde{\Phi}_D$ and $O(b\log|\mathbb{F}|)$ space to maintain $\mathbf{r}^{(1)},\mathbf{r}^{(2)},Q_1$, and Q_2. The prover needs to communicate two degree-S polynomials, which costs $O(S\log|\mathbb{F}|)$. Under the reasonable assumption that the optimal formula for Φ_D depends on all its input variables, we have $S \geq 3b$, which yields a bound of $O(S\log(m+S))$ on the space and help costs. The claim about the runtimes is straightforward from the protocol's description. □

We remark that our above theorem made the tacit *uniformity* assumption that a formula for Φ_D of length $\mathrm{fsize}(D)$ could be constructed in space $O(\mathrm{fsize}(D))$ and time $\mathrm{poly}(\mathrm{fsize}(D))$.

3.3. Range Counting Queries. Let $\mathcal{U}$ be any data universe and $\mathcal{R} \subseteq 2^{\mathcal{U}}$ a set of *ranges*. In the RANGECOUNT problem, the data stream $\sigma = \langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}, R^* \rangle$ specifies a sequence of universe elements $\mathbf{x}^{(i)} \in \mathcal{U}$, followed by a *query* or *target* range $R^* \in \mathcal{R}$. The goal is to output $|\{i : \mathbf{x}^{(i)} \in R^*\}|$, i.e., the number of elements in the target range that appeared in the stream.

We easily obtain a *two-message* streaming interactive proof for the RANGECOUNT problem with cost bounded by $O(\log|\mathcal{R}|\log(\log|\mathcal{R}|+m))$. The verifier simply runs a POINTQUERY on the derived stream σ' defined to have data universe $\mathcal{R}$. σ' is obtained from σ as follows: on each stream update $\mathbf{x}^{(i)} \in \mathcal{U}$, the verifier inserts into σ' one copy of each range $R \in \mathcal{R}$ such that $\mathbf{x}^{(i)} \in R$. The range count problem is equivalent to a POINTQUERY on σ', with the target item being R^*, and we obtain the following theorem.

THEOREM 3.5. *There is a two-message SIP with $O(\log|\mathcal{R}|\log(\log|\mathcal{R}|+m))$ cost for* RANGECOUNT.

The above protocol also implies a *three-message* SIP for the problem of *linear classification*, a core problem in machine learning. Just like the protocol for NEARESTNEIGHBOR invokes a two-message protocol for INDEX, a SIP for linear classification (find a hyperplane that separates red and blue points) verifies that the proposed hyperplane is empty of red points on one side and blue points on the other using the above two-message RANGECOUNT protocol. If the input points lie in $[n]^d$, and $\mathcal{R}$ is the set of halfspaces over $[n]^d$, then $|\mathcal{R}| \leq n^{O(d^2)}$; this bound can be established by combining the Sauer-Shelah lemma [39, 42] with the well-known fact that that VC dimension of halfspaces over $\mathbb{R}^d$ is $d+1$. It follows that the cost of the resulting 3-message SIP over domain $[n]^d$, for streams of length at most m, is $O\big(d^2\log n\log(d\log n+m)\big)$.

The prover and verifier in the protocol of Theorem 3.5 may require time $\Omega(|\mathcal{R}|)$ per stream update. This could be prohibitively large. However, we can obtain savings analogous to Theorem 3.4 if we make a mild "efficient computability" assumption on our ranges. Specifically, suppose there exists a ($\mathrm{poly}(S)$-time uniform) de Morgan formula Φ of length S that takes as input a binary string representing a point $\mathbf{x}^{(i)} \in \mathcal{U}$, as well as the label of a range $R \in \mathcal{R}$ and outputs a bit that is 1 if and only if $\mathbf{x}^{(i)} \in R$. We then obtain the following more practical SIP.

THEOREM 3.6. *Suppose membership in ranges from $\mathcal{R}$ can be decided by de Morgan formulas of length S as above. Then there is a two-message SIP for* RANGECOUNT *on $\mathcal{R}$, with costs at most $O(S\log(m+S))$, in which the verifier runs in time $O(S)$ per stream update, and the prover runs in total time $m \cdot \mathrm{poly}(S)$.*

3.4. Median and Selection Queries. We give a three-message SIP for SELECTION, of which MEDIAN is a special case. In the SELECTION problem, defined over data universe $\mathcal{U} = [n]$, the data stream $\sigma = \langle \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}, \rho \rangle$ is a sequence of elements from $[n]$, followed by a desired rank $\rho \in [m]$. For $i \in [n]$, let $f_i := \{j : \mathbf{x}^{(j)} = i\}$ denote the number of times element i appears in the stream. Given a desired rank $\rho \in [m]$, the goal is to output an element

$j \in [n]$ such that

$$\sum_{k<j} f_k < \rho \quad \text{and} \quad \sum_{k>j} f_k \le m - \rho. \tag{3.2}$$

MEDIAN is the special case of SELECTION when $\rho = \lfloor m/2 \rfloor$.

Our three-message SIPs for SELECTION essentially work by reducing to the RANGE-COUNT problem, but an additional message is required for the prover to send the desired element j to the verifier.

THEOREM 3.7. *There is a three-message SIP for* SELECTION *with cost at most* $O(\log n \log(m + \log n))$ *in which the verifier runs in time* $\mathrm{poly}(\log n, \log m)$ *per update, and the prover runs in total time* $m \cdot \mathrm{poly}(\log n, \log m)$.

Proof. While processing the data stream, the verifier runs two parallel independent instances of a RANGECOUNT protocol with the class of ranges being $\mathcal{R} = \{\{i, i+1, \ldots, n\} : 1 \le i \le n\}$. At the end of the stream, the prover supplies a $j \in [n]$ claimed to satisfy both conditions in Equation (3.2). To check this claim, the verifier need only check that the number of stream elements from the range $\{j, \ldots, n\}$ is at least $m - \rho + 1$, and the number of stream elements from the range $\{j+1, \ldots, n\}$ is at most $m - \rho$. Each check can be performed using one of the invocations of the RANGECOUNT protocol.

Noting that $|\mathcal{R}| = n$, Theorem 3.5 immediately yields a bound of $O(\log n \log(nm))$ on the space and help costs. To obtain the smaller bound in the theorem statement, we use the RANGECOUNT protocol of Theorem 3.6 instead: The claimed costs follow because membership in an interval of the form $\{i, \ldots, n\}$ can be computed by a de Morgan formula of length $O(\log n)$. The SIP uses three messages: one for the prover to send j, and two for the parallel instances of the RANGECOUNT protocol. □

3.5. Pattern Matching Queries. In the pattern matching with wildcards problem, denoted PMW, we are given a stream σ representing text $T = (t_1, \ldots, t_m) \in \{0, 1, *\}^m$ *followed* by a pattern $P = (p_1, \ldots, p_q) \in \{0, 1, *\}^q$. The wildcard symbol $*$ is interpreted as "don't care", and the pattern P is said to occur at location i in t if, for every position j in P, either $p_j = t_{i+j}$ or at least one of p_j and t_{i+j} is the wildcard symbol. The PMW problem is to determine the number of locations at which P occurred in T. PATTERNMATCHING refers to the special case where "don't care" symbols are not permitted. We focus on a binary alphabet; a larger alphabet $\mathcal{U}$ can be handled by replacing each character in $\mathcal{U}$ with its binary representation, growing the parameter q by a factor of $\log |\mathcal{U}|$.

Pattern matching, both with and without wildcards, has been extensively studied within the algorithmic literature, with applications ranging from internet search to computational genetics (see, e.g., [13,25] and the references therein). Verifiable protocols for pattern matching enable searching in the cloud, and complements work on searching in encrypted data within the cloud (e.g., [8]). Cormode et al. [15] described and implemented a SIP for PMW that required roughly $\Theta(\log^2 m)$ messages and had space help costs bounded by $\tilde{\Theta}(\log^2 m)$; concretely, their implementation required well over 1,000 messages, even for quite small streams (of length 2^{17}, say). In stark contrast, our new protocol requires the optimal number of messages: two.

THEOREM 3.8. *There is a 2-message SIP for* PMW *with space and help costs at most* $O(q \log(q + m))$*, in which the verifier runs in time* $O(q)$ *per stream update, and the prover runs in total time* $m \cdot \mathrm{poly}(q)$.

Proof. For concreteness, we begin with a simple protocol for PATTERNMATCHING. The verifier runs the POINTQUERY protocol of Theorem 3.1 on a derived stream σ' defined over a universe of size 2^q, defined as follows. The verifier explicitly stores the most recent q items read in the stream at all times (this requires q bits of space). Since in PATTERNMATCHING there are no wildcards, at each time i, the verifier knows the unique pattern $P^{(i)}$ that the most recent q items are an instance of, and inserts $P^{(i)}$ into σ'. The help and space costs of this protocol are both $O(q\log(q+m))$, and both the prover and verifier run in time $O(q\log(q+m))$ per stream update.

To generalize this to allow wildcards in the pattern and text, the protocol for PMW utilizes a similar approach to the one taken in our NEARESTNEIGHBOR and RANGECOUNT protocols. Essentially, we let $\widetilde{\Phi}$ denote the arithmetization of a circuit that takes as input the binary representation of a pattern $P \in \{0,1,*\}^q$, and the binary representation of q additional symbols from $\{0,1,*\}$, and outputs 1 if and only if the pattern P "matches" the q additional symbols. We then apply the polynomial evaluation protocol to the polynomial $g = \sum_{i\le m} \widetilde{\Phi}^{(i)}$, where $\widetilde{\Phi}^{(i)}$ is the polynomial that indicates whether or not its input "matches" the most recent q stream updates at time i.

In more detail, $\widetilde{\Phi}$ is defined as follows. First define $\phi : \{0,1\}^4 \to \{0,1\}$ and $\Phi : \{0,1\}^{4q} \to \{0,1\}$ as follows:

$$\phi(x_1,x_2,y_1,y_2) = (x_1 \wedge y_1) \vee (\neg x_1 \wedge \neg y_1) \vee x_2 \vee y_2\,,$$

$$\Phi\left(x_{11},x_{12},y_{11},y_{12},\ldots,x_{q1},x_{q2},y_{q1},y_{q2}\right) = \bigwedge_{i=1}^{q} \phi\left(x_{i1},x_{i2},y_{i1},y_{i2}\right).$$

In words, $\phi(x_1,x_2,y_1,y_2) = 1$ iff $x_1 = y_1$ or at least one of x_2 and y_2 equals 1. Furthermore, Φ takes q "blocks" as input, with each block consisting of 4 variables, and outputs 1 iff ϕ evaluates to 1 on every block.

Let $\mathbb{F}$ be a finite field of size at least $8(m+q)$, and let $\widetilde{\phi}$ and $\widetilde{\Phi}$ be the multilinear extensions of ϕ and Φ, respectively, over $\mathbb{F}$. Note that $\widetilde{\Phi}(X_{11},X_{12},Y_{11},Y_{12},\ldots,X_{q1},X_{q2},Y_{q1},Y_{q2}) = \prod_{i=1}^{q} \widetilde{\phi}\left(X_{i1},X_{i2},Y_{i1},Y_{i2}\right)$, and that $\widetilde{\Phi}$ can be evaluated at any point using $O(q)$ field operations.

While processing the stream, the verifier maps each symbol $t_i \in \{0,1,*\}$ read from the stream to a pair of bits (t_{i1},t_{i2}) as follows: $0 \mapsto (0,0), 1 \mapsto (1,0), * \mapsto (0,1)$. At time i, let $P^{(i)}$ denote the $2q$ bits corresponding to the most recent q stream updates. Let $\widetilde{\Phi}^{(i)} : \mathbb{F}^{2q} \to \mathbb{F}$ denote the polynomial $\widetilde{\Phi}$, with the inputs $Y_{11},Y_{12},\ldots,Y_{q1},Y_{q2}$ fixed to the bits in $P^{(i)}$. Let $g : \mathbb{F}^{2q} \to \mathbb{F}$ denote the polynomial

$$g(X_{11},X_{12},\ldots,X_{q1},X_{q2}) := \sum_{1\le i\le m} \widetilde{\Phi}^{(i)}(X_{11},X_{12},\ldots,X_{q1},X_{q2}).$$

The verifier will ultimately apply the polynomial evaluation protocol to g. To do so, the verifier must evaluate $g(\mathbf{r}) = \sum_{i\le m} \widetilde{\Phi}^{(i)}$ while processing the data stream, for a random point $\mathbf{r} \in \mathbb{F}^{2q}$. To accomplish this, the verifier explicitly stores $P^{(i)}$ at all times i, which requires $2q$ bits of space. This enables the verifier to determine $\widetilde{\Phi}^{(i)}$ at all times i, and hence $\mathcal{V}$ can keep a running sum of the $\widetilde{\Phi}^{(i)}(\mathbf{r})$ values.

At the end of the stream, the verifier learns the pattern $P \in \{0,1,*\}^q$, and must output the number of occurrences of P in the text. Let $\mathbf{j} \in \{0,1\}^{2q}$ denote the binary representation of P; then the number of occurrences of P is equal to $g(\mathbf{j})$. Hence, it is sufficient for the verifier to apply the polynomial evaluation protocol to g, thereby evaluating $g(\mathbf{j})$.

The space and help costs are now immediate from the guarantees of Section 2.3 and the fact that g is multilinear (hence its total degree is bounded above by $2q$). The claimed time bounds follows from the fact that each polynomial $\Phi^{(i)}$ can be evaluated at any point in time $O(q)$. □

We remark that the PMW protocol of Theorem 3.8 can be run even if the verifier only knows an upper bound on the length q of the pattern. This is because, for any $q' \le q$, a pattern

$P' \in \{0,1,*\}^{q'}$ is equivalent to the pattern $P \in \{0,1,*\}^q$ obtained from P' by concatenating $q-q'$ wildcard symbols to P'.

4. Communication Protocols and Complexity Classes. We now turn to the study of communication complexity classes motivated by a desire to understand streaming interactive proofs (SIPs) from a complexity-theoretic viewpoint. In this section, we lay out the necessary definitions and terminology to rigorously discuss the notions outlined in Subsection 1.3. In the next section we prove the many parts of Result 1.6.

Communication problems arise naturally out of data stream problems if we suppose Alice holds a prefix of the input stream, and Bob the remaining suffix. The primary goal of such reductions is to obtain space lower bounds on data stream algorithms, so we are free to split the stream at any place we like. For example, most data stream problems in Section 3 are *query problems*, where the input consists of a streamed data set, S, followed by a query, q, to apply to S. In this case, it would be natural to split the input by giving S to Alice and q to Bob.

Communication problems that will play an important role in this paper include the following. The index problem, denoted $\text{INDEX} : \{0,1\}^n \times [n] \to \{0,1\}$ where $[n] := \{1,\ldots,n\}$, is defined via $\text{INDEX}(x,j) = x_j$. The set-intersection and set-disjointness problems INTER, DISJ : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ are defined via $\text{INTER}(x,y) = \neg\text{DISJ}(x,y) = \bigvee_{i=1}^{n}(x_i \wedge y_i)$. Finally, the median relation $\text{MED} : [n]^m \times [n]^m \to [n]$ interprets inputs $x \in [n]^m$ and $y \in [n]^m$ as two halves of a list of numbers, and the valid outputs correspond to the median(s) of the combined list.

Communication Complexity Classes.. All our communication models provide random coins and allow two-sided error probability up to a constant; when unspecified, this constant defaults to $1/3$. Given a communication model $\mathbf{C}$, we denote the corresponding complexity measure of a problem f by $\mathrm{C}(f)$. Following Babai et al. [6], we also denote by $\mathbf{C}$ the corresponding complexity class, defined as the set of all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that $\mathrm{C}(f) = (\log n)^{O(1)}$, i.e., functions that are "easy" in the model $\mathbf{C}$.

We let $\mathbf{R}^{[\mathbf{k},\mathbf{A}]}$ denote the model of randomized communication complexity where Alice and Bob exchange $k \geq 1$ messages in total with Alice sending the first; $\mathbf{R}^{[\mathbf{k},\mathbf{B}]}$ is similar, except that Bob starts. In the **MA** model, the super-player Merlin, who sees all of the inputs, broadcasts a message at the start, following which Alice and Bob run a (two-way, arbitrary-round) randomized "verification" protocol. The $\mathbf{MA}^{[\mathbf{k},\mathbf{A}]}$ and $\mathbf{MA}^{[\mathbf{k},\mathbf{B}]}$ models are restrictions of **MA** where Merlin speaks only to Bob[8] and the verification protocol following Merlin's single message is restricted to lie in $\mathbf{R}^{[\mathbf{k},\mathbf{A}]}$ and $\mathbf{R}^{[\mathbf{k},\mathbf{B}]}$ respectively.

The **MA** model (indeed, its restriction $\mathbf{MA}^{[\mathbf{1},\mathbf{A}]}$) allows us to simulate 1-message SIPs in an obvious way: Merlin sends Bob the prover's message, and Alice sends Bob the verifier's memory contents after it has processed her prefix of the stream. Notice that the order of the two messages is not important, modulo one crucial consideration: Alice must have a private channel to Bob and the random coins used to generate the message from Alice to Bob must be *hidden coins*, invisible to Merlin but shared between Alice and Bob (which is why we called them "hidden coins" rather than "private coins").

The models $\mathbf{OMA}^{[\mathbf{k}]}$, $\mathbf{OIP}^{[\mathbf{k}]}$, and $\mathbf{OIP}_{+}^{[\mathbf{k}]}$, for $k \geq 1$, are obtained by extending $\mathbf{MA}^{[\mathbf{1},\mathbf{A}]}$ to simulate k-message SIP protocols. These communication models work as follows. In each case, Alice and Bob first toss some hidden coins. Then, upon receiving the input, two things happen: (1) Merlin and Bob exchange k messages, with Merlin sending the last message in the interaction, and (2) Alice sends Bob a message, randomized using the hidden coins. After these actions are completed, Bob produces an output in $\{0,1\}$. The differences between the

[8] Our definition breaks symmetry between Alice and Bob because our eventual goal is to study online protocols.

three series of models are as follows.

- In $\mathbf{OMA}^{[\mathbf{k}]}$, (1) happens before (2) and Bob must interact with Merlin before looking at his input. This is directly analogous to $\mathbf{AM}_{\mathsf{TM}}$; see the discussion in Subsection 1.2.
- In $\mathbf{OIP}^{[\mathbf{k}]}$, (1) happens before (2) and Bob may look at his input before talking to Merlin.
- Finally, $\mathbf{OIP}_{+}^{[\mathbf{k}]}$ is like $\mathbf{OIP}^{[\mathbf{k}]}$ except that (2) happens before (1). Thus, Bob's messages may depend on Alice's actual message to Bob, not just on Bob's input and the hidden coins.

In the **AM** model, the parties first choose a public random string, then Merlin broadcasts a message to Alice and Bob, who then run a deterministic communication protocol to arrive at a Boolean output. Since Merlin can in fact predict the exact transcript that Alice and Bob will generate following his message, we can assume without loss of generality that after Merlin's message, Alice and Bob output one bit each indicating whether or not they accept Merlin's prediction.

Cost and Value of Protocols. Let $\mathcal{P}$ be a protocol in a model $\mathbf{C}$ involving Merlin. For each input (x,y), $\mathcal{P}$ defines a game between Merlin and Arthur (recall that Alice and Bob together constitute Arthur), wherein Merlin's goal is to make Arthur output 1. We define the *value* $V^{\mathcal{P}}(x,y)$ to be Merlin's probability of winning this game with optimal play. Given a Boolean function f, we say that $\mathcal{P}$ computes f with *soundness error* ε_s and *completeness error* ε_c if, for all x,y we have

$$(4.1)\qquad f(x,y)=0 \;\Rightarrow\; V^{\mathcal{P}}(x,y)\le\varepsilon_s\,, \quad\text{and}\quad f(x,y)=1 \;\Rightarrow\; V^{\mathcal{P}}(x,y)\ge 1-\varepsilon_c\,.$$

When the above holds with $\varepsilon_c=0$, we say that $\mathcal{P}$ computes f with perfect completeness.

The *verification cost* of $\mathcal{P}$, denoted $\mathrm{vc}(\mathcal{P})$, is the (worst-case) number of bits sent by Alice plus the number of hidden coin tosses; its *help cost* $\mathrm{hc}(\mathcal{P})$ is the number of bits communicated between Merlin and Bob; its communication cost $\mathrm{cc}(\mathcal{P})=\mathrm{hc}(\mathcal{P})+\mathrm{vc}(\mathcal{P})$. For a problem f, we define its complexity $\mathrm{C}(f)=\min\{\mathrm{cc}(\mathcal{Q}) : \mathcal{Q}$ is a $\mathbf{C}$ protocol that solves f with $\max\{\varepsilon_s,\varepsilon_c\}\le 1/3\}$.

Simulating SIPs with Communication Protocols. The observation that a standard (Merlin-free) communication protocol can simulate a data streaming algorithm is the basis for almost all known space lower bounds for streaming algorithms. An analogous simulation motivates our definition of $\mathbf{OIP}^{[\mathbf{k}]}$ and $\mathbf{OIP}_{+}^{[\mathbf{k}]}$ by connecting these models to SIPs.

PROPOSITION 4.1. *Suppose that a function $f(\sigma)$ is computed by a k-message SIP $\mathcal{Q}$ of space cost v and help cost h. Consider the following two-party communication analog f^{cc} of f. Alice's input x specifies a prefix of a stream σ, and Bob's input y specifies the suffix of σ (i.e., σ is the concatenation, $x\circ y$, of x and y). Finally, define $f^{cc}(x,y):=f(\sigma)$. Then f^{cc} is computed by an $\mathbf{OIP}_{+}^{[\mathbf{k}]}$ protocol $\mathcal{P}$ with $\mathrm{vc}(\mathcal{P})\le v$ and $\mathrm{hc}(\mathcal{P})\le h$.*

Suppose further that the messages that the verifier sends to the prover when $\mathcal{Q}$ is run on stream $x\circ y$ are independent of the stream prefix x. Then $\mathcal{P}$ is an $\mathbf{OIP}^{[\mathbf{k}]}$ protocol.

Proof. $\mathcal{P}$ works as follows. Alice simulates running the stream-processing phase of the SIP verifier on x (the prefix of the data stream), and sends to Bob a message m_A describing the state of the algorithm at the end of this simulation. Bob picks up where Alice left off, simulating the SIP verifier's processing of the stream suffix y. At this point, Bob knows that state of the SIP verifier after processing the entire stream σ. Bob is then able to simulate the remainder of the SIP verifier's protocol, treating Merlin as the prover.

The completeness and soundness properties of the SIP $\mathcal{Q}$ imply that $\mathcal{P}$ is an $\mathbf{OIP}_{+}^{[\mathbf{k}]}$ protocol for f^{cc}, and clearly $\mathrm{vc}(\mathcal{P})\le v$ and $\mathrm{hc}(\mathcal{P})\le h$. If the messages that the verifier sends to the prover in $\mathcal{Q}$ are independent of the stream prefix x, then the interaction between Bob and Merlin in $\mathcal{P}$ can occur before Bob receives m_A. In this case, $\mathcal{P}$ is an $\mathbf{OIP}^{[\mathbf{k}]}$ protocol. $\square$

4.1. Relations Among Communication Complexity Classes. We prove a number of inclusion and separation results among our "new" communication complexity classes and relate them to previously studied classes. These are summarized in Figure 1.1, replicated below.

$$\begin{tikzcd}
\mathbf{R}^{[\mathbf{1,A}]} \arrow[d, leftrightarrow] & \mathbf{R}^{[\mathbf{2,B}]} \arrow[d, leftrightarrow] & \mathbf{MA}^{[\mathbf{2,B}]} \arrow[d, leftrightarrow] & \mathbf{AM} \arrow[d, leftrightarrow] & \mathbf{OMA}^{[\mathbf{k}]} \arrow[d, two heads] \\
\mathbf{OIP}^{[\mathbf{1}]} \arrow[r, two heads] \arrow[d, leftrightarrow] & \mathbf{OIP}^{[\mathbf{2}]} \arrow[r, two heads] \arrow[d, two heads] & \mathbf{OIP}^{[\mathbf{3}]} \arrow[r, two heads] & \mathbf{OIP}^{[\mathbf{4}]} \arrow[r, leftrightarrow] & \mathbf{OIP}^{[\mathbf{k}]} \\
\mathbf{OIP}_{+}^{[\mathbf{1}]} & \mathbf{R}^{[\mathbf{3,A}]} \arrow[r] & \mathbf{OIP}_{+}^{[\mathbf{2}]} \arrow[ur] & &
\end{tikzcd}$$

Fig. 4.1: The layout of our communication complexity zoo. An arrow from $\mathbf{C}_1$ to $\mathbf{C}_2$ indicates that $\mathbf{C}_1 \subseteq \mathbf{C}_2$. If the arrow is double-headed, then the inclusion is strict. Within the figure, k is an arbitrary constant larger than 4.

Our results shed light on the landscape of online communication complexity in general.

The simplest online communication model is $\mathbf{R}^{[\mathbf{1,A}]}$, a.k.a. one-way randomized communication. The result $\mathbf{OIP}^{[\mathbf{1}]} = \mathbf{OIP}_{+}^{[\mathbf{1}]} = \mathbf{R}^{[\mathbf{1,A}]}$ establishes that in the world of online communication, introducing Merlin into the model is not enough to obtain super-polynomial efficiency improvements, if *interaction* with Merlin is not permitted. The stronger result that $\mathbf{OMA}^{[\mathbf{k}]} = \mathbf{R}^{[\mathbf{1,A}]}$ for all constants $k > 0$ (this is the full statement of Theorem 5.20) establishes that in the "public coin" setting, the addition of Merlin is not enough to obtain super-polynomial speedups even if interaction with Merlin *is* permitted.

The result that $\mathbf{OIP}^{[\mathbf{2}]} = \mathbf{R}^{[\mathbf{2,B}]}$ (see Corollary 5.7) establishes that in the "hidden coin" setting, the addition of Merlin to the communication model *can* yield super-polynomial efficiency improvements, even if only the barest amount of interaction with Merlin is permitted. However, note that $\mathbf{R}^{[\mathbf{2,B}]}$ is the simplest non-online communication model. Thus the combination of hidden coins and a minimal amount of interaction is enough to simulate only the simplest of the non-online communication protocols.

The result that $\mathbf{OIP}^{[\mathbf{4}]} = \mathbf{OIP}_{+}^{[\mathbf{4}]} = \mathbf{AM}$ (see Corollary 5.14) shows that in the "hidden coin" setting, the addition of Merlin to the communication model permits the simulation even of *non-online interactive proofs*, as soon as Bob is permitted to exchange 4 messages with Merlin.

This in turn explains the somewhat puzzling result that the $\mathbf{OIP}$ and $\mathbf{OIP}_{+}$ hierarchies collapse to the fourth level: both Goldwasser–Sipser [21] and Babai–Moran [7] break down in the $\mathbf{OIP}$ and $\mathbf{OIP}_{+}$ worlds because their transformations *do not preserve online-ness*: they will turn an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol into a "public coin" one, but require Merlin to send a message to Alice. However, as soon as Bob is permitted to exchange 4 messages with Merlin, even online interactive proofs can simulate non-online ones. At this point, the phenomena of classical interactive proofs kick in, and the hierarchies collapse.

5. A Communication Complexity Zoo. We now study our central communication models $\mathbf{OIP}^{[\mathbf{k}]}$ and $\mathbf{OIP}_{+}^{[\mathbf{k}]}$, and prove the web of relationships given in Figure 4.1. Our results are of two types: (1) establishing separations or collapses between levels of the **OIP** and $\mathbf{OIP}_+$ hierarchies, as the case may be, and (2) relating these hierarchies to other previously studied communication complexity classes. We shall first characterize every finite level of the **OIP** hierarchy (the vertical bidirectional arrows in Figure 4.1). Next, in Subsections 5.4 and 5.5, we separate the first four levels of the hierarchy (the horizontal double-headed arrows in the figure). Finally, in Subsection 5.5, we separate the **OIP** and **OMA** hierarchies.

Throughout Section 5, f will denote an arbitrary communication problem given by a Boolean function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, and n will parametrize its "instance size" up to a constant factor, i.e., we will have $\log|\mathcal{X}| + \log|\mathcal{Y}| = \Theta(n)$. We shall use big-$O$ and big-Ω notation to hide constants independent of f, $|\mathcal{X}|$ and $|\mathcal{Y}|$. We shall use the term "ordinary protocol" to mean a randomized communication protocol involving Alice and Bob alone (and no Merlin).

The first level of the hierarchy is easy to characterize.

PROPOSITION 5.1. *We have* $\mathbf{OMA}^{[1]} = \mathbf{OIP}^{[1]} = \mathbf{OIP}_+^{[1]} = \mathbf{MA}^{[1,A]} = \mathbf{R}^{[1,A]}$.

Proof. The definitions immediately show that the first four classes are identical (syntactically) and include $\mathbf{R}^{[1,A]}$, because one can always choose to ignore Merlin. The reverse inclusion $\mathbf{MA}^{[1,A]} \subseteq \mathbf{R}^{[1,A]}$ follows from previous work: Chakrabarti et al. [11] show that for all f we have $\mathrm{R}^{[1,A]}(f) = O\big(\mathrm{MA}^{[1,A]}(f)^2\big)$. ∎

5.1. A Characterization of $\mathbf{OIP}^{[2]}$. The main goal of this subsection is to prove that $\mathbf{OIP}^{[2]} = \mathbf{R}^{[2,B]}$. We start with the following communication result, obtained by combining Theorem 2.2 with Proposition 4.1.

LEMMA 5.2 (Polynomial Evaluation Protocol, Communication Version). *Suppose Alice holds a v-variate polynomial g of total degree d over a field $\mathbb{F}$, and Bob holds a point $\mathbf{j} \in \mathbb{F}^v$. Assume $|\mathbb{F}| > 4d$. Then there is an* $\mathbf{OIP}^{[2]}$ *protocol with communication cost $O((v+d)\cdot \log|\mathbb{F}|)$ for evaluating $g(\mathbf{j})$. Hence,* $\mathrm{OIP}^{[2]}(\textsc{index}) = O(\log n \log\log n)$*, so that* $\textsc{index} \in \mathbf{OIP}^{[2]}$.

The just-proved fact that $\textsc{index} \in \mathbf{OIP}^{[2]}$ is striking: combined with the well-known lower bound $\mathbf{R}^{[1,A]}(\textsc{index}) = \Omega(n)$, it shows that introducing Merlin into the picture while keeping the one-way restriction on the Alice/Bob communication lowers cost exponentially. It is now natural to ask whether $\mathbf{OIP}^{[2]}$ allows such exponential savings for harder problems, such as $\textsc{disj}$. Our next result—a lower bound on $\mathbf{OIP}^{[2]}$ complexity—implies that it does not. Our proof of this result shows how to emulate any $\mathbf{OIP}^{[2]}$ protocol by a standard (i.e., Merlin-less) two-message randomized protocol.

THEOREM 5.3. *Let $\mathcal{P}$ be an* $\mathbf{OIP}^{[2]}$ *protocol computing f. Then* $\mathrm{hc}(\mathcal{P})\,\mathrm{vc}(\mathcal{P}) = \Omega(\mathrm{R}^{[2,B]}(f))$. *In particular,* $\mathrm{OIP}^{[2]}(f) = \Omega\big(\mathrm{R}^{[2,B]}(f)^{1/2}\big)$, *which implies* $\mathbf{OIP}^{[2]} \subseteq \mathbf{R}^{[2,B]}$.

Proof. After appropriate parallel repetition, we may assume that the soundness and completeness errors of $\mathcal{P}$ are at most $1/12$ each. In general, $\mathcal{P}$ takes the following shape: (1) hidden coins are tossed, generating random string r according to distribution $\mathcal{D}$; (2) Bob sends Merlin a message $m_B = m_B(y,r)$; (3) Merlin responds with a message $m_M = m_M(x,y,m_B)$; (4) Alice sends Bob a message $m_A = m_A(x,r)$; (5) Bob outputs a bit given by a function $\mathrm{out}^P(y, m_M, m_A)$.

Let $\mathcal{D}_m$ be $\mathcal{D}$ conditioned on the event $\{m_B(y,r) = m\}$. Note that the distribution $\mathcal{D}_m$ depends on *both* y and m. Since Bob knows y, Bob can determine the distribution $\mathcal{D}_m$ for any value of m (this is not, however, true for Alice, because Alice does not know y).

With this notational setup, we now describe (in Figure 5.1) a two-message ordinary protocol $\mathcal{Q}$ that we claim computes f.

1. Bob samples $\overline{r} \sim \mathcal{D}$, computes $\overline{m} = m_B(y, \overline{r})$, then sends Alice i.i.d. samples $r^{(1)}, \ldots, r^{(h)} \sim \mathcal{D}_{\overline{m}}$, where $h = 36(\mathrm{hc}(\mathcal{P}) + 4)$.
2. Alice sends Bob $m_A\big(x, r^{(1)}\big), \ldots, m_A\big(x, r^{(h)}\big)$.
3. Bob outputs 1 iff $\exists m_M : |\{i \in [h] : \mathrm{out}^{\mathcal{P}}(y, m_M, m_A(x, r^{(i)})) = 1\}| > h/2$.

Fig. 5.1: The $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol $\mathcal{Q}$, which simulates the $\mathbf{OIP}^{[\mathbf{2}]}$ protocol $\mathcal{P}$.

To analyze this protocol, let us first define the *weight* $W_{x,y}(\overline{m})$ of a Bob-message $\overline{m}$ to be the probability that Merlin, playing optimally after receiving $\overline{m}$, convinces Bob to output 1. That is,

$$W_{x,y}(\overline{m}) = \max_{m_M} \Pr_{r \sim \mathcal{D}_{\overline{m}}} \left[\mathrm{out}^{\mathcal{P}}(y, m_M, m_A(x, r)) = 1 \right]. \tag{5.1}$$

Then, with $\overline{m} \sim m_B(y, \mathcal{D})$, the expected weight $\mathbb{E}_{\overline{m}}[W_{x,y}(\overline{m})]$ is at least $11/12$ when $f(x,y) = 1$ and at most $1/12$ when $f(x,y) = 0$.

Correctness on **1***-inputs.* Fix $(x,y) \in f^{-1}(1)$. We shall proceed assuming that the specific Bob-message $\overline{m}$ chosen in Step 1 of $\mathcal{Q}$ satisfies $W_{x,y}(\overline{m}) > 2/3 = 1 - 4(1/12)$; by Markov's inequality, this fails to happen with probability at most $1/4$. Studying (5.1) tell us that there exists a specific Merlin-message m_M^* such that $\Pr_r[\mathrm{out}^{\mathcal{P}}(y, m_M^*, m_A(x,r)) = 1] > 2/3$. Therefore, according to the strategy in Steps 2 and 3, the size of the set $\{i \in [h] : \mathrm{out}^{\mathcal{P}}(y, m_M^*, m_A(x, r^{(i)})) = 1\}$ is a sum of h i.i.d. indicators and exceeds $2h/3$ in expectation. By standard Chernoff bounds (e.g., [33, Theorem 4.4]), the probability that Bob outputs 0 is $2^{-\Omega(h)}$. Thus, overall, the probability that $\mathcal{Q}$ outputs 0 on input (x,y) is at most $1/4 + 2^{-\Omega(h)} < 1/3$.

Correctness on **0***-inputs.* Fix $(x,y) \in f^{-1}(0)$. We shall proceed assuming that the specific Bob-message $\overline{m}$ chosen in Step 1 of $\mathcal{Q}$ satisfies $W_{x,y}(\overline{m}) < 1/3$; by Markov's inequality, this fails to happen with probability at most $1/4$. For each specific Merlin-message m_M, define

$$\mathrm{size}(m_M) = \left| \{i \in [h] : \mathrm{out}^{\mathcal{P}}(y, m_M, m_A(x, r^{(i)})) = 1\} \right|.$$

Then $\mathrm{size}(m_M)$ is a sum of h i.i.d. indicators and has expectation below $h/3$. By standard Chernoff bounds, $\Pr[\mathrm{size}(m_M) > h/2] \le e^{-h/36}$. By a union bound over all possible Merlin-messages m_M, the probability that Bob outputs 1 is at most $2^{\mathrm{hc}(\mathcal{P})} e^{-h/36} < 2^{-4}$, using our choice of h. Adding in the $1/4$ from our Markov argument earlier, the overall probability that $\mathcal{Q}$ outputs 1 on input (x,y) is at most $1/4 + 2^{-4} < 1/3$.

Communication Cost. By definition of the $\mathbf{OIP}^{[\mathbf{2}]}$ model, we have $|r| \le \mathrm{vc}(\mathcal{P})$ and $|m_A| \le \mathrm{vc}(\mathcal{P})$. Thus, each of the two messages in $\mathcal{Q}$ costs at most $h \cdot \mathrm{vc}(\mathcal{P}) = O(\mathrm{hc}(\mathcal{P})\,\mathrm{vc}(\mathcal{P}))$ bits. ∎

The above proof exploits a key property of $\mathbf{OIP}^{[\mathbf{2}]}$ protocols: Bob can sample from the conditional distribution $\mathcal{D}_{\overline{m}}$. This is possible because $m_B = m_B(y,r)$ is independent of Alice's message m_A, a property not satisfied in the stronger $\mathbf{OIP}_+^{[\mathbf{2}]}$ model. This explains why Theorem 5.3 does not apply to $\mathbf{OIP}_+^{[\mathbf{2}]}$, and indeed we shall later give an exponential separation between $\mathbf{OIP}^{[\mathbf{2}]}$ and $\mathbf{OIP}_+^{[\mathbf{2}]}$ in Corollary 5.19.

Theorem 5.3 implies a number of lower bounds for specific problems. We begin with DISJ.

COROLLARY 5.4. *We have* $\Omega(n^{1/2}) \le \mathrm{OIP}^{[2]}(\mathrm{DISJ}) \le O(n^{1/2} \log n)$. *In particular,* DISJ $\notin$ $\mathbf{OIP}^{[\mathbf{2}]}$.

Proof. For the lower bound, we combine Theorem 5.3 with the fact that $\mathrm{R}^{[2,B]}(\textsc{disj}) \geq \mathrm{R}(\textsc{disj}) = \Omega(n)$, the last step being a celebrated lower bound [26]. The upper bound follows from the Aaronson–Wigderson protocol [2] for $\textsc{disj}$, which is in fact an $\mathbf{MA}^{[\mathbf{1},\mathbf{A}]}$ protocol. □

We remark that we may replace $\textsc{disj}$ in Corollary 5.4 with $\textsc{ip2}$, the "inner product mod 2" function. Indeed, the Aaronson–Wigderson protocol also applies to $\textsc{ip2}$, and $\mathrm{R}(\textsc{ip2}) = \Omega(n)$.

Recall that $\textsc{med}$ is a relation on inputs in $[n]^m \times [n]^m$. We next prove a lower bound of $\Omega(m^{1/4})$ on the cost of any $\mathbf{OIP}^{[\mathbf{2}]}$ protocol for $\textsc{med}$. This justifies our use of three messages in the polylogarithmic cost SIP for $\textsc{median}$ we gave in Theorem 3.7, as it implies that any 2-message SIP for median based on known techniques must have polynomial cost.

COROLLARY 5.5. *We have* $\Omega(m^{1/4}) \leq \mathrm{OIP}^{[2]}(\textsc{med}) \leq O(m^{1/2}\log^{3/2} n)$.

Proof. Guha and McGregor [23, Theorem 6.3] gave a reduction from pointer jumping to median computation. Their reduction implies that if $n = \Omega(m^{3/2})$, then $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}(\textsc{med}) = \Omega(m^{1/2})$. The result that $\Omega(m^{1/4}) \leq \mathrm{OIP}^{[2]}(\textsc{med})$ then follows from Theorem 5.3. The upper bound $\mathrm{OIP}^{[2]}(\textsc{med}) \leq O(m^{1/2}\log^{3/2} n)$ follows from an annotated data stream protocol for $\textsc{median}$ given by Chakrabarti et al. [10, Corollary 3.4]. □

We have seen that up to polynomial (specifically, quadratic) blowup, $\mathbf{OIP}^{[\mathbf{2}]}$ is no more powerful than ordinary $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$. We now show that up to another quadratic blowup this is in fact a characterization.

THEOREM 5.6. *For all f, we have* $\mathrm{OIP}^{[2]}(f) = O\big(\mathrm{R}^{[2,B]}(f)^2\big)$. *In particular,* $\mathbf{OIP}^{[\mathbf{2}]} \supseteq \mathbf{R}^{[\mathbf{2},\mathbf{B}]}$.

Proof. Let $\mathcal{Q}$ be an $\mathbf{R}^{[\mathbf{2},\mathbf{B}]}$ protocol for f with cost C and error at most $1/6$. Assume without loss of generality that $C \geq 5$ and that each of the two messages in $\mathcal{Q}$ is a string in $\{0,1\}^C$. We shall treat Alice's messages as elements of the field $\mathbb{F} = \mathbb{F}_{2^C}$ via an agreed-upon bijection.

We design an $\mathbf{OIP}^{[\mathbf{2}]}$ protocol $\mathcal{P}$ for f, based on $\mathcal{Q}$. Given an input (x,y), $\mathcal{P}$ begins by choosing a (hidden) random string r shared between Alice and Bob exactly as $\mathcal{Q}$ would have. From now on, think of x,y,r as fixed. This then fixes a message m_B that Bob would have sent Alice in $\mathcal{Q}$, as well as a function $m_A : \{0,1\}^C \to \mathbb{F}$ specifying Alice's response to each Bob-message. Let $\widetilde{m}_A(Z_1,\ldots,Z_C) \in \mathbb{F}[Z_1,\ldots,Z_C]$ be the multilinear extension of this function m_A. In $\mathcal{P}$, Alice needs to send a message to Bob that allows him to determine $m_A(m_B) = \widetilde{m}_A(m_B)$ with Merlin's help. This is an instance of polynomial evaluation, so we solve it by applying the $\mathbf{OIP}^{[\mathbf{2}]}$ polynomial evaluation protocol (PEP) from Lemma 5.2.

The polynomial $\widetilde{m}_A$ is C-variate and has total degree C. Therefore, PEP has communication cost $O(C\log|\mathbb{F}|) = O(C^2)$, as does $\mathcal{P}$. Next, PEP has perfect completeness, so an honest Merlin can cause $\mathcal{P}$ to output 1 whenever the choice of r would have caused $\mathcal{Q}$ to output 1. Finally, PEP has soundness error at most $C/(|\mathbb{F}|-1) = C/(2^C-1) < 1/6$, so a dishonest Merlin can cause $\mathcal{P}$ to differ in output from $\mathcal{Q}$ with probability at most $1/6$. Using the error bound of $1/6$ on $\mathcal{Q}$, we conclude that $\mathcal{P}$ has completeness error at most $1/6$ and soundness error at most $1/6 + 1/6 = 1/3$. □

COROLLARY 5.7. *For all f, we have* $\Omega\big(\mathrm{R}^{[2,B]}(f)^{1/2}\big) \leq \mathrm{OIP}^{[2]}(f) \leq O\big(\mathrm{R}^{[2,B]}(f)^2\big)$. *Thus,* $\mathbf{OIP}^{[\mathbf{2}]} = \mathbf{R}^{[\mathbf{2},\mathbf{B}]}$.

Proof. Combine Theorems 5.3 and 5.6. □

5.2. A Characterization of $\mathbf{OIP}^{[\mathbf{3}]}$. The main goal of this subsection is to prove that $\mathbf{OIP}^{[\mathbf{3}]} = \mathbf{MA}^{[\mathbf{2},\mathbf{B}]}$. We give a lower bound that builds on the argument in Theorem 5.3. Just as before, we can then derive a lower bound for the specific problem $\textsc{disj}$.

THEOREM 5.8. *Let $\mathcal{P}$ be an* $\mathbf{OIP}^{[3]}$ *protocol computing f. Then there is an* $\mathbf{MA}^{[2,B]}$ *protocol $\mathcal{Q}$ computing f with* $\mathrm{hc}(\mathcal{Q}) \le \mathrm{hc}(\mathcal{P})$ *and* $\mathrm{vc}(\mathcal{Q}) = O(\mathrm{hc}(\mathcal{P})\,\mathrm{vc}(\mathcal{P}))$. *In particular,* $\mathrm{OIP}^{[3]}(f) = \Omega\big(\mathrm{MA}^{[2,B]}(f)^{1/2}\big)$, *which implies* $\mathbf{OIP}^{[3]} \subseteq \mathbf{MA}^{[2,B]}$.

Proof. The high-level idea for building $\mathcal{Q}$ is as follows. After Merlin sends his first message to Bob in $\mathcal{P}$, the remainder of $\mathcal{P}$ is an $\mathbf{OIP}^{[2]}$ protocol. Theorem 5.3 shows how to cut Merlin out of this remaining protocol, replacing it with $\mathbf{R}^{[2,B]}$ protocol. After this replacement, the result is an $\mathbf{MA}^{[2,B]}$ protocol.

In more detail, suppose $\mathcal{P}$ has completeness and soundness errors at most $1/12$. Let $\mathcal{P}_m$ denote the $\mathbf{OIP}^{[2]}$ protocol obtained from $\mathcal{P}$ by fixing Merlin's *first* message to m. Let $\mathcal{Q}_m$ be the $\mathbf{R}^{[2,B]}$ protocol simulating $\mathcal{P}_m$ as in Figure 5.1. Note that $\mathrm{cc}(\mathcal{Q}_m) = O(\mathrm{hc}(\mathcal{P}_m)\,\mathrm{vc}(\mathcal{P}_m))$. Let $\mathcal{Q}$ be the $\mathbf{MA}^{[2,B]}$ protocol where we use $\mathcal{Q}_m$ as Arthur's verification strategy for a message from Merlin. We claim that $\mathcal{Q}$ computes f.

Completeness. Fix $(x,y) \in f^{-1}(1)$. By the completeness of $\mathcal{P}$, there exists some first message m^* from Merlin such that $\mathcal{P}_{m^*}$ outputs 1 with probability at least $11/12$. By the completeness analysis in Theorem 5.3, $\mathcal{Q}_{m^*}$ outputs 1 with probability at least $2/3$. Therefore, if Merlin sends the message m^* in $\mathcal{Q}$, he will cause the output to be 1 with probability at least $2/3$.

Soundness. Fix $(x,y) \in f^{-1}(0)$. By the soundness of $\mathcal{P}$, for every possible first message, m, that Merlin may send, $\mathcal{P}_m$ outputs 1 with probability at most $1/12$. By the soundness analysis in Theorem 5.3, for all m, $\mathcal{Q}_m$ outputs 1 with probability at most $1/3$. Therefore, no matter what Merlin sends as his message in $\mathcal{Q}$, he can cause the output to be 1 with probability at most $1/3$.

Costs. Merlin in $\mathcal{Q}$ sends only part of what Merlin in $\mathcal{P}$ sends; therefore $\mathrm{hc}(\mathcal{Q}) \le \mathrm{hc}(\mathcal{P})$. Furthermore, $\mathrm{vc}(\mathcal{Q}) \le \max_m \mathrm{cc}(\mathcal{Q}_m) = \max_m O(\mathrm{hc}(\mathcal{P}_m)\,\mathrm{vc}(\mathcal{P}_m)) = O(\mathrm{hc}(\mathcal{P})\,\mathrm{vc}(\mathcal{P}))$.

COROLLARY 5.9. *We have* $\Omega(n^{1/3}) \le \mathrm{OIP}^{[3]}(\textsc{disj}) \le O(n^{1/3}\log n)$. *In particular,* $\textsc{disj} \notin \mathbf{OIP}^{[3]}$.

Proof. Klauck [27] proved that $\mathrm{MA}(\textsc{disj}) = \Omega(n^{1/2})$. Applying Theorem 5.8 to this result gives the non-tight bound $\mathrm{OIP}^{[3]}(\textsc{disj}) = \Omega(n^{1/4})$. But we observe that Klauck's proof shows something stronger: namely, if an **MA** protocol $\mathcal{Q}$ computes $\textsc{disj}$, then $\mathrm{hc}(\mathcal{Q})\,\mathrm{vc}(\mathcal{Q}) = \Omega(n)$. Combining Theorem 5.8 with *this* result, we conclude that if an $\mathbf{OIP}^{[3]}$ protocol $\mathcal{P}$ computes $\textsc{disj}$, then $\mathrm{hc}(\mathcal{P})^2\,\mathrm{vc}(\mathcal{P}) = \Omega(n)$, and therefore $\mathrm{hc}(\mathcal{P}) + \mathrm{vc}(\mathcal{P}) = \Omega(n^{1/3})$.

For the upper bound, we note that Aaronson and Wigderson [2] also gave an online **MAMA** protocol for $\textsc{disj}$ of cost $O(n^{1/3}\log n)$. Every online **MAMA** protocol admits a simulation in $\mathbf{OIP}^{[3]}$. ☐

As with Corollary 5.4, we may replace $\textsc{disj}$ in the above result with $\textsc{ip2}$. Indeed, Klauck's result [27] implies that $\mathrm{MA}(\textsc{ip2}) = \Omega(n^{1/2})$, and Aaronson and Wigderson's **MAMA** protocol also applies to $\textsc{ip2}$.

As we did for the second level in the **OIP** hierarchy, we give an upper bound that applies to the third level and gives a characterization that is tight up to a quadratic blowup.

THEOREM 5.10. *For all f, we have* $\mathrm{OIP}^{[3]}(f) = O\big(\mathrm{MA}^{[2,B]}(f)^2\big)$. *In particular,* $\mathbf{OIP}^{[3]} \supseteq \mathbf{MA}^{[2,B]}$.

Proof sketch. We build on the argument in Theorem 5.6 exactly as the proof of Theorem 5.8 builds on Theorem 5.3. Given an $\mathbf{MA}^{[2,B]}$ protocol $\mathcal{Q}$ of cost C, the verification strategy used by Alice and Bob in $\mathcal{Q}$ is an $\mathbf{R}^{[2,B]}$ protocol of cost C, which we can replace with an $\mathbf{OIP}^{[2]}$ protocol of cost $O(C^2)$, by Theorem 5.6. After this replacement we have an $\mathbf{OIP}^{[3]}$ protocol. The remaining analysis is routine. ☐

COROLLARY 5.11. *For all f, $\Omega\big(\mathrm{MA}^{[2,B]}(f)^{1/2}\big) \le \mathrm{OIP}^{[3]}(f) \le O\big(\mathrm{MA}^{[2,B]}(f)^2\big)$. Thus, $\mathbf{OIP^{[3]}} = \mathbf{MA^{[2,B]}}$.*

Proof. Combine Theorems 5.8 and 5.10. ◻

5.3. A Characterization of $\mathbf{OIP^{[4]}}$ and Beyond. The fourth level of the **OIP** hierarchy turns out to have surprising power. It can capture all of **AM**, a model that lies at the frontier of our current understanding of communication complexity classes in the sense that we do not know any nontrivial **AM** lower bounds. Thanks to this surprising power, we can show that all constant-height levels of the **OIP** hierarchy collapse to the fourth level.

PROPOSITION 5.12. *For all f, we have $\mathrm{OIP}^{[4]}(f) = O(\mathrm{AM}(f)\log \mathrm{AM}(f))$. In particular, $\mathbf{OIP^{[4]}} \supseteq \mathbf{AM}$.*

Proof. Suppose $\mathrm{AM}(f) = C$. Without loss of generality, there is a protocol $\mathcal{Q}$ for f with the following shape: Bob tosses coins to generate a random string r and sends it to Merlin, who responds with a message m, where $|r| + |m| \le C$. Bob then sends (r,m) to Alice, who responds with a single bit, after which Bob announces the output.

The interaction between Bob and Alice is an $\mathbf{R^{[2,B]}}$ protocol (in fact, it is deterministic) of cost C. Theorem 5.6 shows that it can be replaced with an $\mathbf{OIP^{[2]}}$ protocol of cost $O(C^2)$. Performing this replacement gives us an $\mathbf{OIP^{[4]}}$ protocol for f. The cost bound can be improved to $O(C\log C)$ by revisiting the analysis of the polynomial evaluation protocol used to prove Theorem 5.6 and using the fact that Alice's message in $\mathcal{Q}$ is just a single bit. ◻

PROPOSITION 5.13. *For each $k > 0$, there exists a constant $c_k > 0$ such that for all f, $\mathrm{OIP}_+^{[k]}(f) \ge \Omega\big(\mathrm{AM}(f)^{c_k}\big)$. In particular, for every constant k, we have $\mathbf{OIP_+^{[k]}} \subseteq \mathbf{AM}$.*

Proof. Let $C = \mathrm{OIP}_+^{[k]}(f)$ and let $\mathcal{P}$ be an $\mathbf{OIP_+^{[k]}}$ protocol with cost C that computes f. By definition, $\mathcal{P}$ uses a hidden random string and Merlin learns about this string only indirectly, from Bob's computed messages. We apply the Goldwasser–Sipser set lower bound technique [21] to convert $\mathcal{P}$ into a protocol where all random coins are directly revealed. Specifically, we can convert $\mathcal{P}$ into an $\mathbf{AMAM}\cdots\mathbf{AM}$ protocol $\mathcal{Q}'$, where $k+3$ messages are sent in total: Merlin's messages are broadcast and after his final message Alice sends a message to Bob, who announces the output. We have $\mathrm{cc}(\mathcal{Q}') = O(C^{a_k})$ for some constant $a_k \ge 1$.

We apply Babai and Moran's round elimination techniques [7] to turn $\mathcal{Q}'$ into a standard **AM** protocol $\mathcal{Q}$ of cost at most $O(\mathrm{cc}(\mathcal{Q}')^{b_k})$ for some constant $b_k \ge 1$. The result follows by taking $c_k = 1/(a_k b_k)$. ◻

This bring us to the main result of this section.

THEOREM 5.14 (Collapse of OIP hierarchy). *For all f, $\Omega\big(\mathrm{AM}(f)^{c_4}\big) \le \mathrm{OIP}^{[4]}(f) \le O(\mathrm{AM}(f)\log \mathrm{AM}(f))$, where c_4 is the constant from Proposition 5.13. In particular, $\mathbf{OIP^{[4]}} = \mathbf{AM}$, and in fact $\mathbf{OIP^{[k]}} = \mathbf{AM}$ for every constant $k \ge 4$.*

Proof. Combine Propositions 5.12 and 5.13, noting that $\mathbf{OIP^{[k]}} \subseteq \mathbf{OIP_+^{[k]}}$ for every $k \ge 4$. ◻

Here is an interesting point worth contemplating. On the one hand, our transformations in the proof of Proposition 5.13 perform round reduction at the expense of destroying online-ness: the final protocol $\mathcal{Q}$ is no longer online, i.e., we cannot require communications to go to Bob alone. On the other hand, the transformation in the proof of Proposition 5.12 "restores" online-ness at only a "slight" expense of requiring Bob and Merlin to exchange four messages, whereas **AM** uses only two. Overall, we have a collapse of the **OIP** hierarchy to its fourth level.

We have also noted earlier (Sections 1.3 and 1.5) that we (regretfully) do not yet know how to place a concrete problem outside $\mathbf{OIP}_{+}^{[2]}$, and subsequent work of Bouland et al. [9] implies that achieving this will require new techniques in communication complexity. Nevertheless, Propositions 5.12 and 5.13 together establish a weakness of $\mathbf{OIP}_{+}^{[2]}$: up to polynomial factors this model is no more powerful than $\mathbf{OIP}^{[4]}$.

5.4. Exponential Separations in Our Complexity Zoo. Among the first four levels of the **OIP** hierarchy, we can now show that every pair of adjacent levels is exponentially separated. The next three results make this precise. Recall that INTER $= \neg$DISJ is the set intersection problem.

THEOREM 5.15. *We have* $\mathrm{OIP}^{[1]}(\text{INDEX}) = \Omega(n^{1/2})$ *whereas* $\mathrm{OIP}^{[2]}(\text{INDEX}) = O(\log n \log\log n)$.

Proof. Combine Proposition 5.1 and Lemma 5.2, and then the known results that $\mathrm{MA}^{[1,A]}(f) = \Omega\big(\mathrm{R}^{[1,A]}(f)^{1/2}\big)$ for all f [11] (see also Theorem 5.20 in Section 5.5), and that $\mathrm{R}^{[1,A]}(\text{INDEX}) = \Omega(n)$ [4]. □

THEOREM 5.16. *We have* $\mathrm{OIP}^{[2]}(\text{INTER}) = \Omega(n^{1/2})$ *whereas* $\mathrm{OIP}^{[3]}(\text{INTER}) = O(\log^2 n)$.

Proof. For the lower bound, use $\mathrm{R}^{[2,B]}(\text{INTER}) \geq \mathrm{R}(\text{INTER}) = \mathrm{R}(\text{DISJ}) = \Omega(n)$ and then apply Theorem 5.3.

For the upper bound, note that INTER has a nondeterministic protocol with cost $O(\log n)$, wherein Alice and Bob guess an element in the intersection of their respective sets and they verify membership. In particular this gives $\mathrm{MA}^{[2,B]}(\text{INTER}) = O(\log n)$; in fact, Bob need not send anything to Alice in the $\mathbf{MA}^{[2,\mathbf{B}]}$ protocol. Now apply Theorem 5.10. □

While we do not know of a *total* Boolean function that separates $\mathbf{OIP}^{[3]}$ from $\mathbf{OIP}^{[4]}$, we do know of a *partial* Boolean function whose $\mathbf{OIP}^{[3]}$ communication complexity is exponentially larger than its $\mathbf{OIP}^{[4]}$ communication complexity. Specifically, Klauck [28, Corollary 3] gives a promise problem he calls PAPPMP which has *Quantum* Merlin-Arthur (**QMA**) communication complexity $\Omega(n^{1/6})$ and **AM** communication complexity $O(\log n)$. Since Theorem 5.8 shows that any $\mathbf{OIP}^{[3]}$ protocol can be transformed into an equivalent $\mathbf{MA}^{[2,\mathbf{B}]}$ protocol with a quadratic blowup in cost, and $\mathbf{MA}^{[2,\mathbf{B}]}$ protocols are simply restricted versions of QMA protocols, Klauck's lower bound on the QMA cost of PAPPMP implies that $\mathrm{OIP}^{[3]}(\text{PAPPMP}) = \Omega(n^{1/12})$.

Meanwhile, Proposition 5.12 shows that any **AM** communication protocol can be transformed into an equivalent $\mathbf{OIP}^{[4]}$ protocol with a logarithmic blowup in costs. Thus, Klauck's upper bound on the **AM** communication complexity of PAPPMP implies that $\mathrm{OIP}^{[4]}(\text{PAPPMP}) = O(\log n \log\log n)$.

THEOREM 5.17. *We have* $\mathrm{OIP}^{[3]}(\text{PAPPMP}) = \Omega(n^{1/12})$ *whereas* $\mathrm{OIP}^{[4]}(\text{PAPPMP}) = O(\log n \log\log n)$.

Next, we show that, up to polynomial factors, $\mathbf{OIP}_{+}^{[2]}$ is at least as powerful as $\mathbf{R}^{[3,\mathbf{A}]}$, the class of *three-message* randomized communication protocols in which Alice speaks first. This will enable us to exhibit an explicit function f on domain $\{-1,1\}^n \times \{-1,1\}^n$ such that $\mathrm{OIP}^{[2]}(f) = \Omega(\sqrt{n/\log n})$, while $\mathrm{OIP}_{+}^{[2]}(f) = O(\log^2 n)$.

THEOREM 5.18. *For all f, we have* $\mathrm{OIP}_{+}^{[2]}(f) = O\big(\mathrm{R}^{[3,A]}(f)^2\big)$.

Proof. Let $\mathcal{Q}$ be any three-message randomized communication protocol of cost C, with Alice speaking first. We show how to convert $\mathcal{Q}$ into an $\mathbf{OIP}_{+}^{[2]}$ protocol $\mathcal{P}$ of cost $O(C^2)$.

We think of $\mathcal{Q}$ as consisting of one message $m_A^{(1)}$ from Alice to Bob, followed by a *two-message* communication protocol $\mathcal{Q}'$ in which Bob speaks first. Theorem 5.6 shows how to transform $\mathcal{Q}'$ into an equivalent $\mathbf{OIP}^{[2]}$ protocol $\mathcal{P}'$ of cost $O(C^2)$ (note this $\mathbf{OIP}^{[2]}$ protocol *depends* on $m_A^{(1)}$).

Thus, we obtain an $\mathbf{OIP}_+^{[2]}$ protocol $\mathcal{P}$ as follows. Alice's message to Bob in $\mathcal{P}$ consists of two parts. The first specifies $m_A^{(1)}$, and the second is the message she would have sent to Bob in $\mathcal{P}'$. Bob, who learns $m_A^{(1)}$ from the first part of Alice's message, now knows what $\mathbf{OIP}^{[2]}$ protocol $\mathcal{P}'$ to execute, and simply behaves the same as he would in $\mathcal{P}'$. $\square$

Exponential separations between $\mathbf{R}^{[3,A]}$ and $\mathbf{R}^{[2,B]}$ are known. In particular, consider the k-step (bipartite) pointer jumping function PJ_k, which interprets each of Alice and Bob's inputs as a list of $N = \Theta(n/\log n)$ *pointers*, a pointer being a $(\log N)$-bit integer. Each pointer in a player's list is interpreted as pointing to (i.e., indexing) a pointer in the other player's list. The goal is to follow these pointers, starting at the first pointer in Alice's list, and output the kth pointer encountered. For example, if Alice's input is $x = (00, 01, 10, 00)$ and Bob's input is $y = (01, 10, 11, 00)$, then $\text{PJ}_1(x,y) = 01$, $\text{PJ}_2(x,y) = 01$, $\text{PJ}_3(x,y) = 10$, and so on. To turn PJ_k into a Boolean function BPJ_k, we take the parity of the $(\log N)$-bit output of PJ_k.

COROLLARY 5.19. *We have* $\text{OIP}^{[2]}(\text{BPJ}_2) = \Omega(\sqrt{n/\log n})$, *while* $\text{OIP}_+^{[2]}(\text{BPJ}_2) = O(\log^2 n)$.

Proof. Nisan and Wigderson [35] showed that $\text{R}^{[k,B]}(\text{BPJ}_k) = \Omega(N/k^2 - k\log N)$. In particular, any two-message randomized communication protocol in which Bob speaks first has cost $\Omega(N)$. Hence, Theorem 5.3 implies that $\text{OIP}^{[2]}(\text{BPJ}_2) = \Omega(\sqrt{n/\log n})$.

To prove the upper bound on $\text{OIP}_+^{[2]}(\text{BPJ}_2)$, note that there is a trivial three-message protocol for PJ_2 (and hence for BPJ_2) of cost $O(\log n)$ in which Alice speaks first. Now apply Theorem 5.18. $\square$

5.5. An Exponential Separation Between $\mathbf{OIP}^{[2]}$ and $\mathbf{OMA}^{[k]}$. In this subsection, we establish that for any function f, $\text{OMA}^{[2k]}(f) = \Omega\big(\text{R}^{[1,A]}(f)^{1/(k+1)}\big)$. An essentially identical lower bound was proven by Klauck and Prakash for a closely related (though not identical) communication model; we provide details for completeness, and in the process identify the crucial details of the communication model that enable the lower bound to hold.

THEOREM 5.20. *For any function f and constant k,* $\text{OMA}^{[2k]}(f) = \Omega\big(\text{R}^{[1,A]}(f)^{1/(k+1)}\big)$.

Proof. We begin by proving the result for the case $k = 1$, showing how to transform any $\mathbf{OMA}^{[2]}$ protocol $\mathcal{P}$ into an $\mathbf{R}^{[1,A]}$ protocol $\mathcal{Q}$ of cost $O(\text{hc}(\mathcal{P})\,\text{vc}(\mathcal{P}))$. This transformation is almost identical to the one of Theorem 5.3, with one crucial change.

Analogously to the proof of Theorem 5.3, $\mathcal{P}$ takes the following shape: (1) hidden coins are tossed, generating random string r according to distribution $\mathcal{D}$; (2) Bob sends Merlin a message $m_B = m_B(r)$; (3) Merlin responds with a message $m_M = m_M(x,y,m_B)$; (4) Alice sends Bob a message $m_A = m_A(x,r)$; (5) Bob outputs a bit given by a function $\text{out}^P(y, m_M, m_A)$.

The key difference between our current setting and that of Theorem 5.3 is that here m_B is a function only of r and not of Bob's input y. The proof of Theorem 5.3 (see Figure 5.1) described a standard protocol $\mathcal{Q}$ in which Bob sends to Alice i.i.d. samples $r^{(1)}, \ldots, r^{(h)} \sim (\mathcal{D} \mid m_B = \overline{m})$, where $h = 36(\text{hc}(\mathcal{P}) + 4)$. In our case, Alice can choose these i.i.d. samples herself, because m_B does not depend on Bob's input y. We therefore obtain an $\mathbf{R}^{[1,A]}$ protocol $\mathcal{Q}$ of cost $O(\text{hc}(\mathcal{P})\,\text{vc}(\mathcal{P}))$, instead of an $\mathbf{R}^{[2,B]}$ protocol as in Theorem 5.3.

The general case proceeds by induction on k. We view an $\mathbf{OMA}^{[2k]}$ protocol $\mathcal{P}$ as an $\mathbf{OMA}^{[2]}$ protocol $\mathcal{P}_1$ followed by an $\mathbf{OMA}^{[2k-2]}$ protocol $\mathcal{P}_2$. Inductively, we can replace $\mathcal{P}_2$ with a $\mathbf{R}^{[1,A]}$ protocol $\mathcal{Q}_2$ of cost $O\left(\text{hc}(\mathcal{P}_2)^{k-1}\,\text{vc}(\mathcal{P}_2)\right) \le O\left(\text{hc}(\mathcal{P})^{k-1}\,\text{vc}(\mathcal{P})\right)$. By concatenating $\mathcal{P}_1$ and $\mathcal{Q}_2$, we obtain an $\mathbf{OMA}^{[2]}$ protocol $\mathcal{P}_3$ for f with $\text{vc}(\mathcal{P}_3) = O\left(\text{hc}(\mathcal{P})^{k-1}\,\text{vc}(\mathcal{P})\right)$

and $\mathrm{hc}(\mathcal{P}_3) \leq \mathrm{hc}(\mathcal{P})$. By our argument in the case $k = 1$, we can transform $\mathcal{P}_3$ into an $\mathbf{R}^{[\mathbf{1},\mathbf{A}]}$ protocol $\mathcal{Q}$ of cost $O\left(\mathrm{hc}(\mathcal{P})^k \,\mathrm{vc}(\mathcal{P})\right)$.

In particular, if $C = \max\{\mathrm{hc}(\mathcal{P}), \mathrm{vc}(\mathcal{P})\}$, then the cost of $\mathcal{Q}$ is $O(C^{k+1})$. This immediately implies that $\mathrm{OMA}^{[2k]}(f) = \Omega\left(\mathrm{R}^{[1,A]}(f)^{1/(k+1)}\right)$, completing the proof. ∎

The main property of the $\mathbf{OMA}^{[\mathbf{k}]}$ communication model exploited in our proof of Theorem 5.20 is the following: in any $\mathbf{OMA}^{[\mathbf{k}]}$ protocol $\mathcal{P}$, for all $i \leq k$, Alice can determine Bob's ith message to Merlin in $\mathcal{P}$ on her own. In particular, the same lower bound would apply to any variant of online Arthur-Merlin communication models in which Bob's messages to Merlin must be independent of his input y. This is the intuitive reason why the $\mathbf{OIP}^{[\mathbf{2}]}$ model is exponentially more powerful than the $\mathbf{OMA}^{[\mathbf{k}]}$ model for any constant k: in the $\mathbf{OIP}^{[\mathbf{2}]}$ model, Bob's message to Merlin may depend on his input y, while this is not allowed in the $\mathbf{OMA}^{[\mathbf{k}]}$ model.

Combining Theorem 5.20 with Lemma 5.2, which says that $\mathrm{OIP}^{[2]}(\textsc{index}) = O(\log n \log\log n)$, we obtain an exponential separation between $\mathbf{OIP}^{[\mathbf{2}]}$ and $\mathbf{OMA}^{[\mathbf{k}]}$ for any constant $k > 0$.

COROLLARY 5.21. *For every constant $k > 0$, we have $\mathbf{OIP}^{[\mathbf{2}]} \not\subseteq \mathbf{OMA}^{[\mathbf{k}]}$.*

6. Conclusion.

Our primary objects of study in this paper were constant-round interactive protocols for verifying outsourced streaming computations. Our main algorithmic contributions were to give constant-round streaming interactive proofs for a large class of "query" problems. Our protocols are exponentially more efficient than what was believed possible based on prior work, and demonstrate that in the streaming setting, "hidden" coins are exponentially more powerful than public coins.

We also introduced new "online" communication hierarchies, $\mathbf{OIP}_+$ and $\mathbf{OIP}$, which can be seen as restricted variants of the standard Arthur-Merlin communication model. The flow of information in the $\mathbf{OIP}_+$ and $\mathbf{OIP}$ models is severely restricted (neither Bob nor Merlin can speak to Alice), yet $\mathbf{OIP}_+$ is still powerful enough to simulate any streaming interactive proof, and $\mathbf{OIP}$ powerful enough to simulate all *known* streaming interactive proofs. Our study revealed that the online nature of these communication models leads them to behave very differently from classical interactive proofs, and allowed us to establish strong limitations on the power of existing techniques for developing constant-round SIPs. It also yielded a surprising characterization of the communication complexity class $\mathbf{AM}$ in terms of online communication models (namely, $\mathbf{AM} = \mathbf{OIP}^{[\mathbf{4}]} = \mathbf{OIP}_+^{[\mathbf{4}]}$). We believe this characterization may prove useful in establishing non-trivial $\mathbf{AM}$ lower bounds, a problem that has been identified [28] as an important "first step" toward resolving the $\mathbf{\Pi_2} \neq \mathbf{\Sigma_2}$ problem in two-party communication complexity, one of the most important problems left open by Babai et al. [6].

Many questions remain for future work, but here we highlight just one: proving a superlogarithmic lower bound on the $\mathbf{OIP}_+^{[\mathbf{2}]}$ communication cost of an explicit function. Progress on this question would yield the first superlogarithmic lower bounds on the cost of two-message SIPs. Moreover, we have shown that standard techniques easily establish that $\mathbf{OIP}_+^{[\mathbf{2}]}$ is a subset of $\mathbf{AM}$, but have been unable to prove any superlogarithmic lower bounds against $\mathbf{OIP}_+^{[\mathbf{2}]}$ protocols. Proving $\mathbf{OIP}_+^{[\mathbf{2}]}$ lower bounds therefore represents an important (and potentially tractable) "zeroth step" toward resolving $\mathbf{\Pi_2} \neq \mathbf{\Sigma_2}$.

REFERENCES

[1] S. AARONSON, *QMA/qpoly $\subseteq$ PSPACE/poly: De-Merlinizing quantum protocols*, in 21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic, 2006, pp. 261–273.

[2] S. AARONSON AND A. WIGDERSON, *Algebrization: A new barrier in complexity theory*, TOCT, 1 (2009).

[3] A. ABDULLAH, S. DARUKI, C. D. ROY, AND S. VENKATASUBRAMANIAN, *Streaming verification of graph properties*, in 27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia, 2016, pp. 3:1–3:14.

[4] F. ABLAYEV, *Lower bounds for one-way probabilistic communication complexity and their application to space complexity*, Theoretical Computer Science, 175 (1996), pp. 139–159.

[5] N. ALON, Y. MATIAS, AND M. SZEGEDY, *The space complexity of approximating the frequency moments*, J. Comput. Syst. Sci., 58 (1999), pp. 137–147.

[6] L. BABAI, P. FRANKL, AND J. SIMON, *Complexity classes in communication complexity theory*, in Proc. 27th Annual IEEE Symposium on Foundations of Computer Science, 1986, pp. 337–347.

[7] L. BABAI AND S. MORAN, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity class*, J. Comput. Syst. Sci., 36 (1988), pp. 254–276.

[8] D. BONEH, G. D. CRESCENZO, R. OSTROVSKY, AND G. PERSIANO, *Public key encryption with keyword search*, in Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, vol. 3027 of Lecture Notes in Computer Science, Springer, 2004, pp. 506–522.

[9] A. BOULAND, L. CHEN, D. HOLDEN, J. THALER, AND P. N. VASUDEVAN, *On the power of statistical zero knowledge*, in Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on, IEEE, 2017, pp. 708–719.

[10] A. CHAKRABARTI, G. CORMODE, N. GOYAL, AND J. THALER, *Annotations for sparse data streams*, in Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2014, pp. 687–706.

[11] A. CHAKRABARTI, G. CORMODE, A. MCGREGOR, AND J. THALER, *Annotations in data streams*, ACM Transactions on Algorithms, 11 (2014), p. 7. A preliminary version of this paper by A. Chakrabarti, G. Cormode, and A. McGregor appeared in *ICALP* 2009.

[12] K. CHUNG, Y. T. KALAI, F. LIU, AND R. RAZ, *Memory delegation*, in Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, 2011, pp. 151–168.

[13] R. CLIFFORD, K. EFREMENKO, E. PORAT, AND A. ROTHSCHILD, *From coding theory to efficient pattern matching*, in Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 778–784.

[14] A. CONDON, *The complexity of space bounded interactive proof systems*, Complexity Theory: Current Research, (1993), pp. 147–190.

[15] G. CORMODE, M. MITZENMACHER, AND J. THALER, *Practical verified computation with streaming interactive proofs*, in Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012, 2012, pp. 90–112.

[16] ———, *Streaming graph computations with a helpful advisor*, Algorithmica, 65 (2013), pp. 409–442.

[17] G. CORMODE, J. THALER, AND K. YI, *Verifying computations with streaming interactive proofs*, PVLDB, 5 (2011), pp. 25–36.

[18] S. DARUKI, J. THALER, AND S. VENKATASUBRAMANIAN, *Streaming verification in data analysis*, in Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings, 2015, pp. 715–726.

[19] S. GOLDWASSER, D. GUTFREUND, A. HEALY, T. KAUFMAN, AND G. N. ROTHBLUM, *A (de)constructive approach to program checking*, in Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08, New York, NY, USA, 2008, ACM, pp. 143–152.

[20] S. GOLDWASSER, Y. T. KALAI, AND G. N. ROTHBLUM, *Delegating computation: Interactive proofs for muggles*, in Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08, New York, NY, USA, 2008, ACM, pp. 113–122.

[21] S. GOLDWASSER AND M. SIPSER, *Private coins versus public coins in interactive proof systems*, in Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC '86, New York, NY, USA, 1986, ACM, pp. 59–68.

[22] M. GÖÖS, T. PITASSI, AND T. WATSON, *The landscape of communication complexity classes*, in 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, 2016, pp. 86:1–86:15.

[23] S. GUHA AND A. MCGREGOR, *Stream order and order statistics: Quantile estimation in random-order streams*, SIAM J. Comput., 38 (2009), pp. 2044–2059.

[24] T. GUR AND R. RAZ, *Arthur-Merlin streaming complexity*, in Automata, Languages, and Programming, F. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg, eds., vol. 7965 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 528–539.

[25] A. KALAI, *Efficient pattern-matching with don't cares*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA., 2002, pp. 655–656.

[26] B. KALYANASUNDARAM AND G. SCHINTGER, *The probabilistic communication complexity of set intersection*, SIAM J. Discret. Math., 5 (1992), pp. 545–557.

[27] H. KLAUCK, *Rectangle size bounds and threshold covers in communication complexity*, in 18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark, 2003, pp. 118–134.

[28] ———, *On Arthur Merlin games in communication complexity*, in Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, June 8-10, 2011, 2011, pp. 189–199.

[29] H. KLAUCK AND V. PRAKASH, *Streaming computations with a loquacious prover*, in Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS '13, New York, NY, USA, 2013, ACM, pp. 305–320.

[30] I. KREMER, N. NISAN, AND D. RON, *On randomized one-round communication complexity*, in Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing, STOC '95, New York, NY, USA, 1995, ACM, pp. 596–605.

[31] S. V. LOKAM, *Spectral methods for matrix rigidity with applications to sizedepth trade-offs and communication complexity*, Journal of Computer and System Sciences, 63 (2001), pp. 449 – 473.

[32] C. LUND, L. FORTNOW, H. KARLOFF, AND N. NISAN, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.

[33] M. MITZENMACHER AND E. UPFAL, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, New York, NY, USA, 2005.

[34] I. NEWMAN AND M. SZEGEDY, *Public vs. private coin flips in one round communication games (extended abstract)*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, 1996, pp. 561–570.

[35] N. NISAN AND A. WIGDERSON, *Rounds in communication complexity revisited*, SIAM Journal on Computing, 22 (1993), pp. 211–219.

[36] Y. OFMAN, *On the algorithmic complexity of discrete functions*, Doklady Akademii Nauk SSSR, 145 (1962), pp. 48–51. English Translation in Soviet Physics Doklady 7: 589-591, 1963.

[37] C. PAPAMANTHOU, E. SHI, R. TAMASSIA, AND K. YI, *Streaming authenticated data structures*, in Advances in Cryptology – EUROCRYPT, Springer, 2013, pp. 353–370.

[38] R. RAZ, *Quantum information and the PCP theorem*, Algorithmica, 55 (2009), pp. 462–489.

[39] N. SAUER, *On the density of families of sets*, Journal of Combinatorial Theory, Series A, 13 (1972), pp. 145–147.

[40] D. SCHRÖDER AND H. SCHRÖDER, *Verifiable data streaming*, in the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012, 2012, pp. 953–964.

[41] A. SHAMIR, *IP = PSPACE*, J. ACM, 39 (1992), pp. 869–877.

[42] S. SHELAH, *A combinatorial problem; stability and order for models and theories in infinitary languages*, Pacific Journal of Mathematics, 41 (1972), pp. 247–261.

[43] J. THALER, *Time-optimal interactive proofs for circuit evaluation*, in Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II, 2013, pp. 71–89.

[44] ———, *Semi-streaming algorithms for annotated graph streams*, in 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, eds., vol. 55 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 59:1–59:14.

[45] V. VU, S. T. V. SETTY, A. J. BLUMBERG, AND M. WALFISH, *A hybrid architecture for interactive verifiable computation*, in 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013, 2013, pp. 223–237.

[46] C. WALLACE, *A suggestion for a fast multiplier*, Electronic Computers, IEEE Transactions on, EC-13 (1964), pp. 14–17.