Interactive Oracle Proofs with Constant Rate and Query Complexity*†

Eli Ben-Sasson[1], Alessandro Chiesa[2], Ariel Gabizon[‡3], Michael Riabzev[4], and Nicholas Spooner[5]

1 **Technion, Haifa, Israel**
eli@cs.technion.ac.il
2 **University of California, Berkeley, CA, USA**
alexch@berkeley.edu
3 **Zerocoin Electronic Coin Company (Zcash), Boulder, CO, USA**
ariel@z.cash
4 **Technion, Haifa, Israel**
mriabzev@cs.technion.ac.il
5 **University of Toronto, Toronto, Canada**
spooner@cs.toronto.edu

Abstract

We study *interactive oracle proofs* (IOPs) [7, 43], which combine aspects of probabilistically checkable proofs (PCPs) and interactive proofs (IPs). We present IOP constructions and techniques that let us achieve tradeoffs in proof length versus query complexity that are not known to be achievable via PCPs or IPs alone. Our main results are:

1. *Circuit satisfiability has* 3*-round IOPs with linear proof length (counted in bits) and constant query complexity.*
2. *Reed–Solomon codes have 2-round IOPs of proximity with linear proof length and constant query complexity.*
3. *Tensor product codes have 1-round IOPs of proximity with sublinear proof length and constant query complexity.* (A familiar example of a tensor product code is the Reed–Muller code with a bound on individual degrees.)

For all the above, known PCP constructions give *quasilinear* proof length and constant query complexity [12, 16]. Also, for circuit satisfiability, [10] obtain PCPs with linear proof length but *sublinear* (and super-constant) query complexity. As in [10], we rely on algebraic-geometry codes to obtain our first result; but, unlike that work, our use of such codes is much "lighter" because we do not rely on any automorphisms of the code.

We obtain our results by building "IOP-analogues" of tools underlying numerous IPs and PCPs:

- **Interactive proof composition.** Proof composition [3] is used to reduce the query complexity of PCP verifiers, at the cost of increasing proof length by an additive factor that is exponential in the verifier's randomness complexity. We prove a composition theorem for IOPs where this additive factor is linear.
- **Sublinear sumcheck.** The sumcheck protocol [34, 46] is an IP that enables the verifier to check the sum of values of a low-degree multi-variate polynomial on an exponentially-large hypercube, but the verifier's running time depends linearly on the bound on individual degrees. We prove a sumcheck protocol for IOPs where this dependence is sublinear (e.g., polylogarithmic).

* A full version of the paper is available at https://eprint.iacr.org/2016/324.

† This work has received funding from the Israel Science Foundation (grant 1501/14), the UC Berkeley Center for Long-Term Cybersecurity, and the United States – Israel Binational Science Foundation (grant 2021036).

‡ This work was done while Ariel Gabizon was at Technion and visiting UC Berkeley.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 40; pp. 40:1–40:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Our work demonstrates that even constant-round IOPs are more efficient than known PCPs and IPs.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases probabilistically checkable proofs, interactive proofs, proof composition, sumcheck

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.40

1 Introduction

We study *Interactive Oracle Proofs* (also known as *Probabilistically Checkable Interactive Proofs*) [7, 43], which combine aspects of probabilistically checkable proofs (PCPs) and interactive proofs (IPs). We present IOP constructions and general techniques that enable us to obtain tradeoffs in proof length versus query complexity that are not known to be achievable by either PCPs or IPs alone. For some applications (e.g., constructing non-interactive arguments in the random oracle model [7]) considering such general types of proof systems suffices (as opposed to focusing only on PCPs or IPs) and thus these applications inherit the efficiency improvements over PCPs.

1.1 Motivation

Probabilistically checkable proofs (PCPs) were introduced by [22, 5, 20, 3, 2]: in a PCP, a probabilistic polynomial-time verifier has oracle access to the proof string. The complexity class $\mathbf{PCP}[\mathsf{r}, \mathsf{q}]$ denotes those languages for which the verifier uses at most r random bits and queries at most q proof locations; the proof length is then at most 2^{r}. The PCP Theorem [3, 2] states that $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$: every NP statement has a proof of polynomial length that can be verified via a constant number of queries (say, with soundness error $1/2$).

A fundamental question is how long a PCP needs to be, compared to the corresponding "standard" NP proof. Given $T\colon \mathbb{N} \to \mathbb{N}$, the PCP Theorem states that every language $\mathcal{L}$ in $\mathbf{NTIME}(T)$ has a proof of length $\mathrm{poly}(T(n))$ that can be verified with $O(1)$ queries. A sequence of works [42, 30, 24, 13, 9, 12, 16] gradually reduced the proof length to quasilinear, i.e., $T(n) \cdot \mathrm{polylog}(T(n))$; much of this progress was accompanied by progress on efficient reductions from **NTIME** to "PCP-friendly" problems, as well as efficient constructions of PCPs of proximity (PCPPs) for key classes of linear codes. Despite much progress, the following question remains open: *are there PCPs with linear proof length and constant query complexity?*

Ben-Sasson et al. [10] make progress in this direction by proving that there is $a > 0$ such that for every $\epsilon > 0$ there is a PCP for circuit satisfiability with proof length $2^{a/\epsilon} n$ and query complexity n^{ϵ}. Beyond the sublinear query complexity, [10]'s result comes with other caveats not affecting most prior constructions: the verifier is *non-uniform*, namely it requires a polynomial-size advice string for every circuit size; and the verifier is not succinct, namely it cannot run in time that is sublinear in the circuit size even if the circuit comes from a uniform circuit family. (Recent constructions of high-rate locally testable codes with sub-polynomial query complexity [32] are not yet known to be convertible to PCPs with similar parameters.)

In this paper, we continue the study of the tradeoff between proof length and query complexity, but we do so for a natural extension of the PCP model (sufficient for some useful

applications, e.g., [7]) that can be thought of as a "multi-round PCP", described below. Also, from this point onwards, we switch to using relations instead of languages. We denote by $\mathcal{R}$ a relation consisting of pairs $(\mathtt{x}, \mathtt{w})$, where $\mathtt{x}$ is the *instance* and $\mathtt{w}$ is the *witness*; we think of $\mathcal{R}$ naturally induced by a non-deterministic language $\mathcal{L}$. We denote by $\mathcal{R}|_{\mathtt{x}}$ the (possibly empty) set of witnesses for a given instance $\mathtt{x}$, and by n the size of $\mathtt{x}$.

1.2 A more general model: interactive oracle proofs

Interactive Oracle Proofs (IOPs) are a type of proof system introduced in [7, 43] that combines aspects of IPs [4, 26] and PCPs [5, 3, 2], and generalizes interactive PCPs [31]. IOPs naturally extend the notion of a PCP to multiple rounds or, viewed from an another angle, they naturally extend the notion of an IP by allowing probabilistic checking. Prior work shows that IOPs can be used to construct non-interactive proofs in the random oracle model [7], that IOPs efficiently achieve unconditional zero knowledge [6], and that IOPs can be used to obtain doubly-efficient constant-round IPs for polynomial-time bounded-space computations [43].

Informally, an IOP extends an IP as follows: whenever the prover sends to the verifier a message, the verifier does not have to read the message in full but may probabilistically query it. In more detail, a k-round IOP comprises k rounds of interaction. In the i-th round of interaction: the verifier sends a message m_i to the prover; then the prover replies with a message f_i to the verifier, which the verifier can query in this and all later rounds (by having oracle access to it). After the k rounds of interaction, the verifier either accepts or rejects.

An *IOP system* for a relation $\mathcal{R}$ with round complexity k and soundness ε is a pair (P, V), where P, V are probabilistic algorithms, that satisfies natural notions of completeness and soundness: for every instance-witness pair $(\mathtt{x}, \mathtt{w})$ in $\mathcal{R}$, $V(\mathtt{x})$ always accepts after $\mathsf{k}(n)$ rounds of interaction with $P(\mathtt{x}, \mathtt{w})$; and, for every instance $\mathtt{x}$ with $\mathcal{R}|_{\mathtt{x}} = \emptyset$ and unbounded prover $\tilde{P}$, $V(\mathtt{x})$ accepts with probability at most $\varepsilon(n)$ after $\mathsf{k}(n)$ rounds of interaction with $\tilde{P}$.

Like the IP model, one efficiency measure is the round complexity k. Like the PCP model, two additional efficiency measures are the *proof length* l, which is the total number of alphabet symbols in all of the prover's messages, and the *query complexity* q, which is the total number of locations queried by the verifier across all of the prover's messages. Considering all of these parameters, we say that a relation $\mathcal{R}$ belongs to the complexity class $\mathbf{IOP}[\mathsf{k}, \mathsf{a}, \mathsf{l}, \mathsf{r}, \mathsf{q}, \varepsilon]$ if there is an IOP system for $\mathcal{R}$ in which on instances of size n:

1. the number of rounds is $\mathsf{k}(n)$;
2. the prover messages are over the alphabet $\mathsf{a}(n)$;
3. the proof length over this alphabet is $\mathsf{l}(n)$;
4. the verifier uses $\mathsf{r}(n)$ random bits;
5. the verifier queries the prover messages in $\mathsf{q}(n)$ locations;
6. the soundness error is $\varepsilon(n)$.

Many other definitions for IPs and PCPs carry over naturally. An IOP is *public coin* if m_i is a random string and the verifier postpones any oracle queries until after receiving all the oracles from the prover (i.e., after the k-th round of interaction). An IOP is *non-adaptive* if the query locations do not depend on answers to any previous queries.

Prior work on IOPs. In prior work, [7] prove that public-coin IOPs can be compiled into non-interactive proofs in the random oracle model; their compiler is as a generalization of the Fiat–Shamir paradigm for public-coin IPs [21, 41], and of the "CS proof" constructions of Micali [37] and Valiant [49] for PCPs. Also, [6] construct 2-round IOPs (called "duplex

PCPs" there) with unconditional zero knowledge and quasilinear proof length; in comparison, short PCPs with unconditional zero knowledge are not known. Also, [43] use IOPs to obtain doubly-efficient constant-round IPs for polynomial-time bounded-space computations. In this paper, we do not study compilers for cryptographic proofs, nor zero knowledge, nor applications to interactive proofs; instead, we focus on tradeoffs of proof length versus query complexity for IOPs.

Prior work on interactive PCPs. An interactive PCP [31] is a PCP followed by a standard IP; in particular, it is an IOP where the verifier sends an empty first message and may query only the first prover message (but must read any other prover messages in full). Prior work on interactive PCPs obtains proof length that depends on the witness size rather than computation size [31, 25], as well as unconditional zero knowledge [28]. In this paper we also study proof length but our results to not seem to extend to the more restricted setting of interactive PCPs.

1.3 Proximity and robustness

To facilitate upcoming technical discussions we briefly introduce two notions that strengthen a PCP.

- *PCPs of proximity* (PCPPs) [17, 9]. On the one hand, a PCP verifier has oracle access to a candidate proof π and only decides if $\mathcal{R}|_{\mathtt{x}} \neq \emptyset$ ($\mathtt{x} \in \mathcal{L}$) or $\mathcal{R}|_{\mathtt{x}} = \emptyset$ ($\mathtt{x} \notin \mathcal{L}$). On the other hand, a PCPP verifier has oracle access to a candidate witness $\mathtt{w}$ and proof π and decides if $\mathtt{w} \in \mathcal{R}|_{\mathtt{x}}$ or $\mathtt{w}$ is far from $\mathcal{R}|_{\mathtt{x}}$ (in particular, if $\mathcal{R}|_{\mathtt{x}} = \emptyset$, then $\mathtt{w}$ is far from $\mathcal{R}|_{\mathtt{x}}$). A quantity δ known as the *proximity parameter* specifies what "far" means: if $\mathtt{w}$ is δ-far from $\mathcal{R}|_{\mathtt{x}}$ then the PCPP verifier accepts with probability at most ε, where ε is the soundness error.
- *Robust PCPs* [9]. When $\mathcal{R}|_{\mathtt{x}} = \emptyset$, the answers to the verifier's queries are, with high probability, far from any answers that make the verifier accept. A quantity ρ known as the *robustness parameter* specifies what "far" means: if $\mathcal{R}|_{\mathtt{x}} = \emptyset$ then, with probability at least $1 - \varepsilon$, the answers are ρ-far from accepting ones.

The two above notions can also be combined, yielding the definition of a *robust PCP of proximity*.

Extension to IOPs. The notions of proximity and robustness naturally extend to IOPs; see the full version for details. For example, we say that an IOP has *proximity parameter* δ if the analogous property for PCPs of proximity holds; we can then correspondingly define the complexity class $\mathbf{IOPP}[\mathsf{k}, \mathsf{a}, \mathsf{l}, \mathsf{r}, \mathsf{q}, \varepsilon, \delta]$.

2 Results

We obtain several IOP constructions with proof length and query complexity that are not known to be achievable either via PCPs or IPs alone (or even via interactive PCPs [31]). First, we show that for circuit satisfiability we can obtain IOPs with linear proof length and constant query complexity; constant round complexity and public coins suffice.

▶ **Theorem 1** (informal). *Let $\mathcal{R}$ be the relation consisting of instance-witness pairs (ϕ, w) where ϕ is a boolean circuit (of two-input NAND gates) and w is a binary input that satisfies ϕ; we use n to denote the number of gates in ϕ. There exists $a > 0$ and a public-coin IOP*

system that puts $\mathcal{R}$ in the complexity class

$$\mathbf{IOP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(n) & = & 3 \\ \text{answer alphabet} & \mathsf{a}(n) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(n) & = & a \cdot n \\ \text{query complexity} & \mathsf{q}(n) & = & a \\ \text{soundness error} & \varepsilon(n) & = & 1/2 \end{array}\right].$$

In particular, via [40]'s reduction from Turing machines to circuits, we deduce that

$$\mathbf{NTIME}(T) \subseteq \mathbf{IOP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(T) & = & 3 \\ \text{answer alphabet} & \mathsf{a}(T) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(T) & = & a \cdot T \log T \\ \text{query complexity} & \mathsf{q}(T) & = & a \\ \text{soundness error} & \varepsilon(T) & = & 1/2 \end{array}\right].$$

The main points of comparison of the above theorem with prior work are the following.

- For PCPs with constant query complexity, prior work achieved only quasilinear proof length [12, 16], with the "quasilinear" hiding several logarithmic factors. In comparison, we achieve linear proof length for circuit satisfiability, and $O(T \log T)$ proof length for nondeterministic T-time relations.
- Ben-Sasson et al. [10] show that there is $a > 0$ such that for every $\epsilon > 0$ there is a non-uniform PCP for circuit satisfiability with proof length $2^{a/\epsilon}n$ and query complexity n^{ϵ}; the non-uniformity comes from the use of algebraic-geometry (AG) codes with transitive automorphism groups, for which uniform families are not known. In comparison, we simultaneously achieve linear proof length and constant query complexity; moreover, we make a much "lighter" use of AG codes, which also allows us to avoid non-uniformity. Namely, we rely only on the multiplication properties of AG codes [14, 36], and do not rely on any code automorphisms. Looking ahead, this is because we do not route circuits on Cayley graphs induced by the automorphisms of the underlying code, unlike [10].

Second, we show that Reed–Solomon codes over binary fields (fields of characteristic 2) have 2-round IOPs of proximity with linear proof length and constant query complexity. Such codes are a key ingredient for constructing PCPs with quasilinear proof length [12]. Recall that a word $w\colon D \to \mathbb{F}$ is represented via $|w| = |D| \cdot \log|\mathbb{F}|$ bits.

▶ **Theorem 2** (informal). *Given a "fractional degree" $\varrho \in (0,1)$, define $\mathcal{R}$ to be the relation consisting of instance-witness pairs $((\mathbb{F}_{2^\lambda}, d), w)$ where $d \leq \varrho 2^\lambda$ and $w\colon \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ is the evaluation of a polynomial of degree less than d; we define the instance size to be λ, and note that w has $|w| = 2^\lambda \cdot \lambda$ bits. For every $\delta \in (0, \frac{1}{2}(1-\varrho))$ there exist $a > 0$ and a public-coin IOP of proximity (P, V) that puts $\mathcal{R}$ in the complexity class*

$$\mathbf{IOPP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(\lambda) & = & 2 \\ \text{answer alphabet} & \mathsf{a}(\lambda) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(\lambda) & = & a \cdot 2^\lambda \cdot \lambda \\ \text{query complexity} & \mathsf{q}(\lambda) & = & a \\ \text{soundness error} & \varepsilon(\lambda) & = & 1/2 \\ \text{proximity parameter} & \delta(\lambda) & = & \delta \end{array}\right].$$

More generally, our result concerns *additive Reed–Solomon codes*, where the domain of a codeword is a λ-dimensional affine subspace S of a potentially larger binary field $\mathbb{F}$; in such cases the above statement involves more parameters but achieves the same asymptotics. The

main point of comparison of the above theorem with prior work is [12, 16], who achieve PCPs of proximity with the same parameters but superlinear proof length: $a \cdot 2^{\lambda} \cdot \lambda \cdot \mathrm{poly}(\lambda)$.

Third, we show that tensor product codes have 1-round IOPs of proximity with sublinear proof length and constant query complexity. Given a positive integer m and linear code C with domain D and alphabet $\mathbb{F}$, the tensor product code $C^{\otimes m}$ is the linear code that comprises all functions $w\colon D^m \to \mathbb{F}$ whose restriction to any axis-parallel line is in C; the message length, block length, and distance of $C^{\otimes m}$ are each the m-th power of the corresponding parameters of C. Tensor product codes are a large family, and they include Reed–Muller codes (at least when considering the definition that bounds the variables' individual degrees, which we do, as opposed to the one that bounds their sum).

▶ **Theorem 3** (informal). *Let $m \geq 3$ and C be a linear code with domain D, alphabet $\mathbb{F}$, and relative distance τ; let $\ell := |D|$ be the block length. Define $\mathcal{R}$ to be the relation of instance-witness pairs $\big((C, m), w\big)$ such that $w \in C^{\otimes m}$; note that w has $|w| = \ell^m \cdot \log|\mathbb{F}|$ bits. For every $\delta \in (0, \frac{1}{2}\tau^m)$ there exist $a > 0$ and a public-coin IOPP system (P, V) that puts $\mathcal{R}$ in the complexity class*

$$\mathbf{IOPP}\left[\begin{array}{lccl} \text{rounds} & \mathsf{k}(\ell^m) & = & 1 \\ \text{answer alphabet} & \mathsf{a}(\ell^m) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(\ell^m) & = & o(\ell^m \cdot \log|\mathbb{F}|) \\ \text{query complexity} & \mathsf{q}(\ell^m) & = & a \\ \text{soundness error} & \varepsilon(\ell^m) & = & 1/2 \\ \text{proximity parameter} & \delta(\ell^m) & = & \delta \end{array}\right].$$

The main points of comparison of the above theorem with prior work are the following.

- Ben-Sasson and Sudan [11] and Viderman [51] give local testers for all tensor product codes with query complexity $\mathsf{q}(\ell^m) = \ell^2$; Dinur et al. [18] give local testers with $\mathsf{q}(\ell^m) = \ell$ for certain tensor product codes. In contrast, we achieve constant query complexity, with only sublinear proof length, for all tensor product codes. Moreover, given additional mild conditions, we obtain constant soundness error *even for non-constant m*.
- The work of [12, 16] implies PCPs of proximity for tensor product codes with superlinear proof length and constant query complexity. In contrast, we obtain sublinear proof length, with a single round of interaction.

Analogously to [51], we can invoke Theorem 3 on different choices of linear codes so to derive different code families that have good properties and an IOP tester (instead of a local tester as in [51]). For example, we can choose a family of linear codes with arbitrarily high rate, constant relative distance, linear-time encoding, and linear-time decoding from a constant fraction of errors [48, 29, 44]; our theorem implies a code with the same properties that *also* has a 1-round IOP of proximity with sublinear proof length and constant query complexity (cf. [51, Section 3.1]).

Similar statements hold for list-decodable codes with good parameters [27] (cf. [51, Section 3.2]); and also for locally correctable and, more generally, locally decodable codes with good parameters [52, 50, 19, 33, 32] (cf. [51, Section 3.3]). In each of these cases, the tensor product operation preserves the "key" properties of the choice of underlying code C, while endowing the resulting code with an IOP of proximity.

We obtain the above results via techniques of independent interest: we prove that, in the IOP model, there are more efficient analogues of tools that are fundamental to constructing PCPs and IPs. We now discuss these techniques.

3 Techniques

Recall that IOPs generalize both IPs, by treating the prover's messages as oracle strings, and PCPs, by allowing for multiple rounds of interaction; they also generalize interactive PCPs [31]. We prove that IOPs can express two fundamental techniques in a more efficient way than in these prior models:

(i) in *interactive proof composition*, the prover is more efficient than in PCP proof composition; and

(ii) in *sublinear sumchecks*, the verifier is more efficient than in IP sumcheck protocols.

We now discuss both of our new tools, and then how we use them.

3.1 Interactive proof composition

Proof composition [3] is used to reduce PCP query complexity, cf. [2, 30, 9]; it involves two PCPs: an outer one and an inner one. One should think of the outer proof system as having short proofs but large query complexity, while the inner proof system has long proofs but small query complexity.

The composed prover uses the outer prover to send a PCP to the composed verifier, who does not run the outer verifier but, instead, uses the inner verifier to check that the outer verifier would have accepted had it made its queries to the PCP. The composed verifier also needs an auxiliary sub-PCP for the claim that the outer verifier would have accepted; in fact, he needs one sub-PCP for each possible random string of the outer verifier. Hence, the composed prover also sends all of these sub-PCPs along with the first PCP. The benefit is that the query complexity of the composed verifier equals that of the inner verifier, which is typically verifying a much smaller statement than the outer verifier.

Beyond query complexity, most other parameters of the composed proof system are simply the sum of corresponding parameters of the outer and inner proof systems. An exception is the proof length l: it does not simply equal the sum $\mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}}$, but instead equals $\mathsf{l}_{\mathsf{out}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot \mathsf{l}_{\mathsf{in}}$, because the composed prover uses the inner proof system to generate a proof *for each choice of randomness of the outer proof system*. (The same is true for prover running time.)

We prove an **Interactive Proof Composition Theorem** that avoids the above limitations. The outer proof system is a robust PCP $(P_{\mathsf{out}}, V_{\mathsf{out}})$ for a relation $\mathcal{R}$, while the inner one is a k-round IOP $(P_{\mathsf{in}}, V_{\mathsf{in}})$ for V_{out}'s relation; the composed proof system is a $(\mathsf{k}+1)$-round IOP (P, V) for $\mathcal{R}$. The parameters of the composed proof system are exactly as before, except that now the new proof length is *much smaller*: $\mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}}$. (Ditto for the prover running time.) The crucial observation is that, after the prover sends the outer proof to the verifier, *soundness is not harmed if the verifier tells the prover his choice of outer randomness*; hence, the prover does not have to invest work for all randomness choices but can simply invest work only for the outer randomness that was chosen, which he now knows.

▶ **Theorem 4** (Interactive Proof Composition – informal). *Suppose that the relation $\mathcal{R}$ satisfies the following:*

(1) there exists a robust PCPP system $(P_{\mathsf{out}}, V_{\mathsf{out}})$ that puts $\mathcal{R}$ in the complexity class

$$\textbf{PCPP}\left[\begin{array}{ll} \text{proof length} & \mathsf{l}_{\mathsf{out}} \\ \text{randomness} & \mathsf{r}_{\mathsf{out}} \\ \text{query complexity} & \mathsf{q}_{\mathsf{out}} \\ \text{soundness error} & \varepsilon_{\mathsf{out}} \\ \text{proximity parameter} & \delta_{\mathsf{out}} \\ \text{robustness parameter} & \rho_{\mathsf{out}} \end{array}\right]$$

and

(2) for every $\mathtt{x}$ there exists an IOPP system $(P_{\mathsf{in}}, V_{\mathsf{in}})$ that puts V_{out}'s relation in the complexity class

$$\textbf{IOPP}\left[\begin{array}{ll} \text{rounds} & \mathsf{k}_{\mathsf{in}} \\ \text{proof length} & \mathsf{l}_{\mathsf{in}} \\ \text{randomness} & \mathsf{r}_{\mathsf{in}} \\ \text{query complexity} & \mathsf{q}_{\mathsf{in}} \\ \text{soundness error} & \varepsilon_{\mathsf{in}} \\ \text{proximity parameter} & \delta_{\mathsf{in}} \end{array}\right]$$

If $\delta_{\mathsf{in}} \leq \rho_{\mathsf{out}}$ then there exists an IOPP system (P, V) that puts $\mathcal{R}$ in the complexity class

$$\textbf{IOPP}\left[\begin{array}{lcl} \text{rounds} & \mathsf{k} & = 1 + \mathsf{k}_{\mathsf{in}} \\ \text{proof length} & \mathsf{l} & = \mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}} \\ \text{randomness} & \mathsf{r} & = \mathsf{r}_{\mathsf{out}} + \mathsf{r}_{\mathsf{in}} \\ \text{query complexity} & \mathsf{q} & = \mathsf{q}_{\mathsf{in}} \\ \text{soundness error} & \varepsilon & = \varepsilon_{\mathsf{out}} + \varepsilon_{\mathsf{in}} \\ \text{proximity parameter} & \delta & = \delta_{\mathsf{out}} \end{array}\right].$$

The above discussion and informal theorem statement omit many technical details that already arise in non-interactive proof composition (e.g., see lengthy discussions in [9, 8]), and we also need to deal with. For instance, one has to clarify the size of the sub-claim on which the the inner proof system is invoked; also, one has to carefully define the notion of a verifier to allow for the composed verifier's running time to be *smaller* than the outer verifier's query complexity. For more details, see the full version.

3.2 Sublinear sumcheck

The *sumcheck protocol* [34, 46] is an interactive proof for the claim "$\sum_{\vec{\alpha} \in H^m} w(\vec{\alpha}) = 0$", where w is the evaluation on $\mathbb{F}^m$ of an m-variate polynomial of individual degree d and H is a subset of $\mathbb{F}$. More generally, w may be a codeword in the tensor product code $C^{\otimes m}$, for a given linear code C with domain D and alphabet $\mathbb{F}$, and H is a subset of D [36]. The prover receives H and w as input, while the verifier receives H as input and w as an oracle. The protocol has m rounds and, if C has relative distance τ, the protocol has soundness error $1 - \tau^m$; also, the prover runs in time $\text{poly}(\ell^m)$, and the verifier in time $\text{poly}(\ell + m)$, where $\ell := |D|$ is C's block length.

In each round, the verifier receives a codeword w_i in C and checks that $\sum_{\alpha \in H} w_i(\alpha)$ equals a certain value γ_{i-1} determined in the previous round; in particular, the verifier reads $\Omega(\ell)$ bits. We show that the verifier complexity can be *sublinear* in ℓ, if the prover and verifier engage in an IOP instead of an IP. The intuition to "go sublinear" is simple: instead of doing these checks explicitly, the verifier uses proximity testers for doing so. Thus, in each round, the prover sends to the verifier two oracles: the codeword in w_i, and a proximity proof attesting that $w_i \in C$ and that $\sum_{\alpha \in H} w_i(\alpha) = \gamma_{i-1}$. The use of proximity proofs complicates the soundness analysis because the verifier only sees noisy codewords, but the backbone of the proof follows that of the standard sumcheck protocol. Overall, we obtain a sumcheck IOP protocol that enables a verifier to efficiently check sumchecks for codes of much larger blocklength than what he can afford in the standard sumcheck protocol.

We state our **Sublinear Sumcheck Theorem** below as a reduction: given a PCP of proximity $(P_{\text{SC}}, V_{\text{SC}})$ for subcodes of the form $C|_{H,\gamma} := \{w \in C \text{ s.t. } \sum_{\alpha \in H} w(\alpha) = \gamma\}$, we

construct an IOP of proximity (P, V) for sumchecks over H^m for $C^{\otimes m}$. The complexity of the PCPP verifier V_{SC} determines the complexity of the resulting IOPP verifier V; e.g., if the former is sublinear in C's block length ℓ, so is the latter.

▶ **Theorem 5** (Sublinear Sumcheck – informal). *Let m be a positive integer, and C a linear code with relative distance τ and block length ℓ. Suppose that there is a PCP of proximity for subcodes of the form $C|_{H,\gamma} := \{w \in C \text{ s.t. } \sum_{\alpha \in H} w(\alpha) = \gamma\}$ with proof length l_{SC}, query complexity q_{SC}, soundness error $\varepsilon_{\mathrm{SC}}$, proximity parameter δ_{SC}, prover running time $\mathsf{tp}_{\mathrm{SC}}$, and verifier running time $\mathsf{tv}_{\mathrm{SC}}$. Then there is a public-coin IOP for sumchecks over H^m for $C^{\otimes m}$ with the following parameters:*

$$\mathbf{IOP}\left[\begin{array}{llll} \text{rounds} & \mathsf{k} & = & m \\ \text{proof length} & \mathsf{l} & = & m \cdot \mathsf{l}_{\mathrm{SC}} + m \cdot \ell \\ \text{query complexity} & \mathsf{q} & = & m \cdot \mathsf{q}_{\mathrm{SC}} + m + 1 \\ \text{soundness error} & \varepsilon & = & 1 - \tau^m + \left(\varepsilon_{\mathrm{SC}} + m \cdot \delta_{\mathrm{SC}}\right) \\ \text{prover time} & \mathsf{tp} & = & m \cdot \mathsf{tp}_{\mathrm{SC}} + m \cdot \ell^m \\ \text{verifier time} & \mathsf{tv} & = & m \cdot \mathsf{tv}_{\mathrm{SC}} + O(m) \end{array}\right].$$

In later sections, it is more natural to state the theorem without assuming that w is a codeword in $C^{\otimes m}$, so the reduction also takes as input a PCP of proximity $(P_{\otimes}, V_{\otimes})$ for $C^{\otimes m}$ that is invoked on w; this introduces additional terms in the parameters. More generally, both of the PCPs of proximity $(P_{\mathrm{SC}}, V_{\mathrm{SC}})$ and $(P_{\otimes}, V_{\otimes})$ can in fact be IOPs of proximity, and we state our theorem for this more general case, which we need. For more details, see the full version.

3.3 Applying the new tools

We now sketch how we use the above new tools to derive the results of Section 2. We begin by discussing our results on proximity testing to codes (stated later); we then turn to circuit satisfiability (stated earlier) because its proof requires one of these results on proximity testing.

Intuition behind Theorem 2. The construction of linear-size IOPs of proximity for Reed–Solomon codes over binary fields follows from one invocation of our Interactive Proof Composition Theorem with [12]'s robust PCPs of proximity for Reed–Solomon codes as the outer proof system, and [38]'s PCPs of proximity for nondeterministic languages as the inner proof system. Informally, in the first round, the prover sends to the verifier a [12] PCP of proximity, which reduces proximity testing of codewords over $\mathbb{F}_{2^\lambda}$ to proximity testing of sub-codewords over $\mathbb{F}_{2^{\lambda/2}+O(1)}$ with only constant overheads; in the second round, the verifier sends his choice of outer randomness, and the prover replies with a [38] PCP of proximity for the sub-codeword. The proof length of this latter component is quasilinear, but is applied to a claim of "square-root size" only, so we obtain linear proof length.

Intuition behind Theorem 3. The construction of sublinear-size IOPs of proximity for tensor product codes follows from one invocation of our Interactive Proof Composition Theorem with [11, 51]'s robust local tester for tensor product codes as the outer proof system, and [38]'s PCPs of proximity for nondeterministic languages as the inner proof system. Unlike before, we now use one round, because the outer proof system only relies on a local tester rather than a PCP of proximity. The verifier thus simply sends his choice of outer randomness, and the prover replies with a [38] PCP of proximity for a suitable sublinear-size

sub-codeword. Since the proof length of this latter component is quasilinear but is applied to a sublinear-size claim, we obtain sublinear proof length.

A summary: overall, we can summarize the above sketches via the following diagram of implications.

Theorem 2 linear-size IOPP for Reed–Solomon codes	⟵	Theorem 4 interactive proof composition	+	[12] robust PCPs of proximity for Reed–Solomon codes	+	[38] PCP of proximity for **NTIME**
Theorem 3 sublinear-size IOPP for tensor product codes	⟵	Theorem 4 interactive proof composition	+	[11, 51] robust local testing for tensor product codes	+	[38] PCP of proximity for **NTIME**

Intuition behind Theorem 1. We now turn to how to construct 3-round IOPs for circuit satisfiability with linear proof length and constant query complexity.

The first step of many PCP constructions is to arithmetize the NP statement at hand (in our case, the satisfiability of a boolean circuit) by reducing it to a "PCP-friendly" problem that looks like a constraint satisfaction problem over a well-chosen graph and whose assignments involve codewords in a well-chosen linear code C. Meir observes in [35, 36] that key features of C are good relative distance and, moreover, a *multiplication property*: coordinate-wise multiplication of codewords yields codewords in a code whose relative distance is still good [14, 36]. Moreover, to obtain short PCPs, the aforementioned graph is typically chosen so to behave like a routing network [42]; for example, [12] use De Bruijn graphs, while [10] use hypercubes. To support such graphs, the automorphism group of C has to be rich enough. This typically holds for Reed–Solomon codes [12] which have a doubly-transitive automorphism group, but is a significantly harder condition to fulfill for AG codes [10], for which obtaining a transitive automorphism group is quite involved and, currently, can only be achieved non-uniformly.

The aforementioned first step would be problematic in our setting, because known routing techniques introduce either logarithmic overheads (as in [12]) or large query complexity (as in [10]), so it is not clear how we could use them. Departing from these prior works, we do not rely on any routing, and instead immediately leverage one round of interaction to directly reduce circuit satisfiability to a sumcheck instance over a given linear code C. Also, we only assume that C has good relative distance and a multiplication property [14], *but we do not rely on any automorphisms.*

Informally, the prover first sends three codewords w_1, w_2, w_3 over a field $\mathbb{F}$; the first codeword encodes values of the left wires of all gates, the second encodes values for the right wires of all gates, and the third encodes values for the output wires of all gates. (When a gate has fan-out greater than 1 we still consider 1 output wire.) The verifier now must check several things. First, that wire values are boolean and the output gate wire equals 0. Second, that the wire values are "locally consistent" with each gate: for every $i \in [n]$, $w_3(i)$ is the NAND of $w_1(i)$ and $w_2(i)$. Third, that the three encodings of wire values are consistent with the circuit topology: namely, if $\ell(i)$ represents the left wire used to compute i, and $r(i)$ represents the right wire used to compute i, the topology requires that $w_3(\ell(i)) = w_1(i)$ and $w_3(r(i)) = w_2(i)$ for every i. The verifier cannot directly conduct these checks (as doing so would incur linear query complexity); instead, the verifier sends some randomness to the prover so to "bundle" the checks into one sumcheck.

But how should the verifier sample randomness to achieve this bundling? One option is to sample a random element in $\mathbb{F}$ per check so to construct a random subset sum, which can be viewed as an n-variate polynomial of total degree 1, whose coefficients are the checks,

evaluated at a random point. If not all checks are satisfied, the polynomial is non-zero, and its random evaluation cannot attain any value with too large probability. However, constructing a random subset sum is inefficient because the verifier samples and sends to the prover $\Omega(n)$ random bits, in order to describe the random point. Nevertheless, the verifier may hope to do better by using a *different* low-degree polynomial for the bundling. In general, if the polynomial has m variables each of degree at most d, the verifier must sample and send m field elements; this preserves soundness provided that $|\mathbb{F}| = \Omega(md)$ (for a constant probability of avoiding any particular output value by the Schwartz–Zippel Lemma [45, 53, 15]) and $d^m = \Omega(n)$ (to bundle all checks). For example, the univariate case of $m = 1$ was considered in [5] when reducing to a sumcheck problem; the multivariate case of $m = \log n$ or $m = \frac{\log n}{\log \log n}$ was considered in later works. Unfortunately, either setting does not work for *constant-size* fields, which we ultimately use to obtain linear proof length.

Taking a step back from polynomials, we see that all we need is an *evading set* S for $\mathbb{F}^n$, which is a small set such that for any non-zero $v \in \mathbb{F}^n$ the inner product $\langle r, v \rangle$, for random $r \in S$, does not attain any particular value $a \in \mathbb{F}$ with too high probability. Good constructions of evading sets are known: they relax a well-studied notion called ϵ-biased sets [39]. In particular, results of [1] imply that, for any ϵ, $\mathbb{F}^n$ has an evading set S of size $\text{poly}(\frac{n}{\epsilon})$ and the aforementioned probability is $\gamma := \epsilon + \frac{1}{|\mathbb{F}|}$; in particular, such a construction is suitable for constant-size fields.

Below we informally state the reduction (see the full version for details), using the following notion: we say that a linear code C' is a *degree d-closure* of C if, for every $w_1, \ldots, w_m \in C$ and m-variate polynomial P of total degree at most d, it holds that $w' \in C'$ where the i-th entry of w' is the evaluation of P on the i-th coordinates of $w_1, \ldots, w_m$.

▶ **Lemma 6** (Circuit SAT to Sumcheck – informal). *Let n be a positive integer, $C \subseteq \mathbb{F}^D$ an n-systematic linear code, ϕ an n-gate boolean circuit (of two-input NAND gates), and S an evading set for $\mathbb{F}^n$. There is a 1-round IOP that reduces satisfiability of ϕ to proximity testing to C and a sumcheck over any degree-3 closure of C. Moreover, the IOP introduces only constant overheads in all relevant parameters, including proof length and query complexity.*

After reducing circuit satisfiability to sumcheck over the given code C, we are left to choose C so to ensure that the sumcheck can be carried out with 2 additional rounds, linear proof length, and constant query complexity.

For this, our starting point is [23, 47]'s efficient construction of a code family with constant rate, relative distance, and alphabet size. Note that since these codes are AG codes, they have a naturally-defined degree-3 closure. Also, their construction is uniform, and thus represents a much "lighter" use of AG codes as compared to in [10].

If we simply choose C to be a code from this AG code family, then it is not clear how to efficiently conduct the sumcheck. However, what does work is to take C to be the tensor product of $O(1)$ copies of this AG code. Informally, in this way, we can invoke our Sublinear Sumcheck Theorem (Theorem 5) on the tensor product code C and we can test proximity to it by Theorem 3. See the full version for details.

Overall, we can summarize the above sketch via the following diagram of implications.

$$\underset{\substack{\text{linear-size IOP}\\ \text{for circuit SAT}}}{\textbf{Theorem 1}} \longleftarrow \underset{\substack{\text{from circuit SAT}\\ \text{to sumcheck}}}{\text{Lemma 6}} + \underset{\substack{\text{sublinear}\\ \text{sumcheck}}}{\text{Theorem 5}} + \underset{\substack{\text{sublinear-size IOP}\\ \text{for tensor product codes}}}{\text{Theorem 3}} + \underset{\substack{\text{efficient construction}\\ \text{of AG codes}}}{[23, 47]}$$

4 Open questions

The question of whether there exist PCPs with linear proof length and constant query complexity remains open. Nevertheless, our work suggests additional questions that may be stepping stones in this and other intriguing directions:

1. Is there a *one*-round IOP for circuit satisfiability with linear proof length and query complexity? (Our IOP for circuit satisfiability requires 3 rounds.)
2. Is there an IOP for $\mathbf{NTIME}(T)$ with linear proof length and query complexity, for *some* number of rounds? (Our results, like [10], only imply proof length $O(T \log T)$.)
3. Is there an IOP for *succinct* circuit satisfiability with linear proof length and query complexity? (Our results, like [10], "stop" at **NP** but do not extend to **NEXP**.)

Finally, while "positive" applications of IOPs are known (e.g., non-interactive proofs in the random oracle model [7]), "negative" ones are not: do IOP constructions with good parameters imply inapproximability results that are not known to be implied by known PCP constructions?

References

1 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS'92.

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS'92.

4 László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC'85, pages 421–429, 1985.

5 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC'91, pages 21–32, 1991.

6 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasilinear-size zero knowledge from linear-algebraic PCPs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC'16-A, pages 33–64, 2016.

7 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, TCC'16-B, pages 31–60, 2016.

8 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC'05, pages 120–134, 2005.

9 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

10 Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate PCPs for Circuit-SAT with sublinear query complexity. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'13, pages 320–329, 2013.

11 Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Structures and Algorithms*, 28(4):387–402, 2006.

12 Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC'05.

13 Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC'03, pages 612–621, 2003.

14 D. V. Chudnovsky and G. V. Chudnovsky. Algebraic complexities and algebraic curves over finite fields. *Journal of Complexity*, 4(4):285–316, 1988.

15 Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

16 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

17 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'04, pages 155–164, 2004.

18 Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, and of the 10th International Workshop on Randomization and Computation*, APPROX-RANDOM'06, pages 304–315, 2006.

19 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012. Preliminary version appeared in STOC'09.

20 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS'91, pages 2–12, 1991.

21 Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO'86, pages 186–194, 1986.

22 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. In *Theoretical Computer Science*, pages 156–161, 1988.

23 Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996.

24 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53:558–655, July 2006. Preliminary version in STOC'02.

25 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC'08, pages 113–122, 2008.

26 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC'85.

27 Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. *SIAM Journal on Computing*, 40(5):1432–1462, 2011. Preliminary version appeared in STOC'09.

28 Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pages 173–190, 2010.

29 Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005. Preliminary version appeared in STOC'03.

30 Prahladh Harsha and Madhu Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3–4):157–201, Dec 2000. Preliminary version in STACS'01.

31 Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP'08, pages 536–547, 2008.

32 Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally-correctable and locally-testable codes with sub-polynomial query complexity. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC'16, pages 202–215, 2016.

33 Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM*, 61(5):28:1–28:20, 2014. Preliminary version appeared in STOC'11.

34 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

35 Or Meir. Combinatorial PCPs with short proofs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, CCC'12, 2012.

36 Or Meir. IP = PSPACE using error-correcting codes. *SIAM Journal on Computing*, 42(1):380–403, 2013.

37 Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS'94.

38 Thilo Mie. Short PCPPs verifiable in polylogarithmic time with o(1) queries. *Annals of Mathematics and Artificial Intelligence*, 56:313–338, 2009.

39 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC'90, pages 213–223, 1990.

40 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

41 David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 387–398, 1996.

42 Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC'94, pages 194–203, 1994.

43 Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC'16, pages 49–62, 2016.

44 Ron M. Roth and Vitaly Skachek. Improved nearly-MDS expander codes. *IEEE Transactions on Information Theory*, 52(8):3650–3661, 2006.

45 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

46 Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

47 Kenneth W. Shum, Ilia Aleshnikov, P. Vijay Kumar, Henning Stichtenoth, and Vinay Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert–Varshamov bound. *IEEE Transactions on Information Theory*, 47(6):2225–2241, 2001.

48 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996. Preliminary version appeared in STOC'95.

49 Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC'08, pages 1–18, 2008.

50 Michael Viderman. A note on high-rate locally testable codes with sublinear query complexity, 2010. ECCC TR10-171.

51 Michael Viderman. A combination of testability and decodability by tensor products. *Random Structures and Algorithms*, 46(3):572–598, 2015. Preliminary version appeared in APPROX-RANDOM'12.

52 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1), 2008. Preliminary version appeared in STOC'07.

53 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the 1979 International Symposium on Symbolic and Algebraic Computation*, EUROSAM'79, pages 216–226, 1979.