A Hierarchy Theorem for Interactive Proofs of Proximity

Tom Gur[*1] **and Ron D. Rothblum**[†2]

1 **Weizmann Institute, Rehovot, Israel**
tom.gur@weizmann.ac.il
2 **MIT, Cambridge, USA**
rothblum@gmail.com

Abstract

The number of rounds, or round complexity, used in an interactive protocol is a fundamental resource. In this work we consider the significance of round complexity in the context of *Interactive Proofs of Proximity* (IPPs). Roughly speaking, IPPs are interactive proofs in which the verifier runs in sublinear time and is only required to reject inputs that are far from the language.

Our main result is a round hierarchy theorem for IPPs, showing that the power of IPPs grows with the number of rounds. More specifically, we show that there exists a gap function $g(r) = \Theta(r^2)$ such that for every constant $r \geq 1$ there exists a language that (1) has a $g(r)$-round IPP with verification time $t = t(n, r)$ but (2) does not have an r-round IPP with verification time t (or even verification time $t' = \mathsf{poly}(t)$).

In fact, we prove a stronger result by exhibiting a *single* language $\mathcal{L}$ such that, for every constant $r \geq 1$, there is an $O(r^2)$-round IPP for $\mathcal{L}$ with $t = n^{O(1/r)}$ verification time, whereas the verifier in *any* r-round IPP for $\mathcal{L}$ must run in time at least t^{100}. Moreover, we show an IPP for $\mathcal{L}$ with a poly-logarithmic number of rounds and only poly-logarithmic verification time, yielding a sub-exponential separation between the power of constant-round IPPs versus general (unbounded round) IPPs.

From our hierarchy theorem we also derive implications to standard interactive proofs (in which the verifier can run in polynomial time). Specifically, we show that the round reduction technique of Babai and Moran (JCSS, 1988) is (almost) optimal among all blackbox transformations, and we show a connection to the algebrization framework of Aaronson and Wigderson (TOCT, 2009).

1998 ACM Subject Classification F.1.3 [Computation by Abstract Devices] Complexity Measures and Classes

Keywords and phrases Complexity Theory, Property Testing, Interactive Proofs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.39

1 Introduction

Interactive Proofs, introduced by Goldwasser at el. [38] (and in their public-coin form, by Babai and Moran [8]), are protocols in which a computationally unbounded prover tries to convince a verifier that an input x belongs to a language $\mathcal{L}$. A recent line of work, initiated by Ergün, Kumar and Rubinfeld [19] and more recently by Rothblum, Vadhan and

[*] Tom Gur is supported by the ISF grant number 671/13 and Irit Dinur's ERC grant number 239985; part of this research was conducted while visiting Columbia University, New York.

[†] Ron Rothblum is partially supported by NSF MACS - CNS-1413920 and by SIMONS Investigator award Agreement Dated 6-5-12.

8th Innovations in Theoretical Computer Science Conference (ITCS 2017).
Editor: Christos H. Papadimitrou; Article No. 39; pp. 39:1–39:43
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Wigderson [58], considers a variant of interactive proofs in which the verifier is required to run in *sublinear* time. Since the verifier does not have enough time to even read its entire input, we cannot expect it to reject every false statement. Rather, following the property testing literature [59, 27] (see also [26]), we relax the soundness condition and only require that the verifier reject inputs that are *far* from the language (no matter what cheating strategy the prover uses). Since the verifier is only assured that the input is close to the language, such interactive proofs are called *interactive proofs of proximity* (IPPs). Indeed, IPPs may be thought of as the property testing analogue of interactive proofs.

From an information theoretic perspective, the key parameters of an IPP are its *query complexity*, *communication complexity* and *round complexity.* The *query complexity* is the number of bits of the input string that the verifier reads. The *communication complexity* is the number of bits exchanged between the prover and the verifier, and the *round complexity* is the number of rounds of interaction. We think of all of these parameters as being *sublinear* in the input length. Additional computational parameters that we aim to minimize are the verifier's running time (which should also be sublinear) and the prover's running time (which, ideally, should be proportional to the complexity of deciding the language).

In this work we focus on the round complexity of IPPs, and on the relation between the number of rounds and the other parameters. Specifically, we ask the following question:

Does the power of Interactive Proofs of Proximity grow with the number of rounds?

Understanding the round complexity of protocols is a central problem in the theory of computation (most notably in complexity theory and cryptography). Some of the main motivations for reducing round complexity are considerations such as network latency, the need to stay online or to synchronize messages between the parties, and the overhead involved in sending and receiving messages.

1.1 Our Results

Our main result answers the foregoing question by showing a hierarchy of IPPs: we show that for a gap function $g(r) = \Theta(r^2)$, and for every constant $r \geq 1$, it holds that r-round IPPs can be outperformed by $g(r)$-round IPPs, in the sense that the verifier in the latter system is significantly more efficient. We prove our hierarchy theorem by constructing a *single* explicit language for which the power of IPPs grows with the number of rounds.

▶ **Theorem 1** (Hierarchy theorem, informally stated (see Theorem 16)). *There exists an explicit language $\mathcal{L}$ such that for every constant $r \geq 1$ and for inputs of length n:*

1. *There is an $O(r^2)$-round* IPP *for $\mathcal{L}$ in which the verifier runs in time $t = n^{O(1/r)}$; and*
2. *The verifier in* any *r-round* IPP *for $\mathcal{L}$* must *run in time at least $t' = t^{100}$ (where the constant 100 is arbitrary). Furthermore, either the communication complexity or query complexity of the verifier must be at least t'.*

Thus, we obtain a characterization (which is exact, up to the specific polynomial of the gap function g) of the complexity of constant-round IPPs for the language $\mathcal{L}$.

For simplicity, the statement in Theorem 1 is restricted to constant-round protocols. However, the complexity of the IPP protocol in Theorem 1 actually reduces further as the round complexity grows to be super-constant. In particular, we obtain a poly-logarithmic round IPP for $\mathcal{L}$ with poly-logarithmic communication and query complexities, and an $\omega(1)$-round IPP with $n^{o(1)}$ communication and query complexities. Together with the lower bound in Theorem 1, these yield a separation between the power of constant-round IPPs and super-constant round IPPs, and a *sub-exponential* separation with respect to poly-logarithmic round IPPs.

▶ **Theorem 2** (Constant Round versus General IPPs)**.** *There exists a language $\mathcal{L}$ that has a* $\mathsf{polylog}(n)$*-round* IPP *with a* $\mathsf{polylog}(n)$ *time verifier and an* $\omega(1)$*-round* IPP *with* $n^{o(1)}$ *time verifier, but for every constant* $r \geq 1$*, the verifier in any* r*-round* IPP *for* $\mathcal{L}$ *must run in time at least* $n^{\Omega(1/r)}$*.*

Prior to this work, only a separation between the power of MAPs (which are *non-interactive* IPPs, i.e., the entire "interaction" consists of a *single* message) and IPPs was known [43].

We remark that Theorems 1 and 2, and their proofs, shed new light also on standard interactive proofs (in which the verifier is given direct access to the input and can run in polynomial time). We proceed to discuss such implications.

Optimality of the Babai-Moran Round Reduction. Following Vadhan [66], we consider *black-box transformations on interactive proofs*, which are transformations that take prover and verifier strategies $(\mathcal{P}, \mathcal{V})$, for an interactive-proof for some language $\mathcal{L}$, and output new strategies $(\mathcal{P}', \mathcal{V}')$, for the same language $\mathcal{L}$, such that new prover and verifier strategies can only make oracle calls to the original strategies $(\mathcal{P}, \mathcal{V})$. More specifically, the new verifier $\mathcal{V}'$ is only allowed to make oracle calls to $\mathcal{V}$ (and in particular does not have direct access to the input) and $\mathcal{P}'$ may make oracle calls to both $\mathcal{V}$ and $\mathcal{P}$.[1]

As pointed out by Vadhan, many (but not all) of the known transformations on interactive proofs from the literature are in fact black-box. We focus on such a transformation, due to Babai and Moran [8], for reducing the number of rounds of interaction in public-coin interactive proofs. Using our hierarchy theorem, we show that the overhead incurred by the round reduction transformation of [8] is close to optimal among all black-box transformations.

Algebrization of Interactive Proofs. As our second application, we show a connection between our hierarchy theorem and the *algebrization* framework of Aaronson and Wigderson [1]. This framework, which is an extension of the *relatization* framework of Baker, Gill, and Solovay [9], is viewed as a barrier to proving complexity-theoretic lower bounds using currently known proof techniques. Loosely speaking, [1] show that almost all known complexity theoretic results "algebrize" (i.e., fall within their framework), whereas making progress on some of our most fundamental questions (such as $\mathcal{P} \neq \mathsf{NP}$) requires non-algebrizing techniques.

Using our hierarchy theorem for IPPs, we show that any proof of the complexity class inclusion $\#\mathcal{P} \subseteq \mathcal{AM}$ (which is widely disbelieved, and in particular implies the collapse of the polynomial hierarchy) must make use of non-algebrizing techniques, and therefore *must* introduce a fundamentally different proof technique. A conceptual connection between our results and interactive proofs in the algebrization framework is further discussed in Section 1.3 and elaborated on in Section 5.

1.2 Technical Overview

Loosely speaking, the language for which we prove the round hierarchy theorem consists of error-correcting encodings of strings $x \in \{0,1\}^k$ whose Hamming weight $\mathsf{wt}(x) \overset{\text{def}}{=} \sum_{i \in [k]} x_i$, is divisible by 3 (i.e., $\mathsf{wt}(x) = 0 \pmod 3$).

[1] One could also restrict $\mathcal{P}'$ to make only oracle calls to $\mathcal{P}$ (and not to $\mathcal{V}$ as we do). However, giving $\mathcal{P}'$ more freedom only makes our results stronger (since we rule out the broader class of transformations).

The specific encoding that we use is the low degree extension code $\mathsf{LDE} : \mathbb{F}^k \to \mathbb{F}^n$, over a field $\mathbb{F}$ that is an extension field of $\mathsf{GF}(2)$.[2] Indeed, it is crucial that the characteristic of $\mathbb{F}$ is different than the modulus 3. The parameter k (which specifies the message length) is the same as in the preceding paragraph, where we view $\{0,1\}^k$ as a subset of the message space $\mathbb{F}^k$.

Before proceeding, we note that throughout this work we use the standard convention that codes map messages of length k to codewords of length $n = \mathsf{poly}(k)$. In particular, this will mean that inputs to IPPs, which will typically refer to (possibly corrupt) codewords, have length n, whereas inputs to other types of protocols and sub-routines, may refer to the underlying messages, which have length k.

Recall that the LDE code is parameterized by a finite field $\mathbb{F}$, a subset of the field $H \subseteq \mathbb{F}$ and a dimension m. To encode a message $x \in \{0,1\}^k$, where $k = |H|^m$, we view the message as a function $x : H^m \to \{0,1\}$ (by associating $[k]$ with H^m) and consider the unique individual degree $|H|-1$ polynomial $P : \mathbb{F}^m \to \mathbb{F}$ that agrees with x on H^m. We denote this polynomial by $P = \mathsf{LDE}_{\mathbb{F},H,m}(x)$. For the time being, the sizes of $|\mathbb{F}|$, $|H|$ and m should all be thought of as at most poly-logarithmic in n. (See Section 2.3 for additional details about the LDE encoding.)

Thus, the language for which we prove our round hierarchy, which we denote by Enc-MOD3, consists of all polynomials $P : \mathbb{F}^m \to \mathbb{F}$ of individual degree $|H|-1$ that obtain Boolean values in the subcube H^m, such that these Boolean values sum up to $0 \pmod 3$. That is, all polynomials P such that $P|_{H^m} : H^m \to \{0,1\}$ and $\sum_{z \in H^m} P(z) \equiv 0 \pmod 3$.

We prove our hierarchy theorem by showing that for every constant $r \geq 1$, the language Enc-MOD3 has an $O(r^2)$-round IPP in which the verifier runs in time roughly $n^{O(1/r)}$, and that the verifier in *any* r-round IPP for Enc-MOD3 must run in time at least $n^{\Omega(1/r)}$, where the constant in the Ω-notation can be made arbitrarily larger than the constant in the O-notation. In Section 1.2.1 we give an overview of the upper bound, which is technically more involved, and then, in Section 1.2.2 we give an overview of the lower bound.

1.2.1 Upper Bound

Our goal is to construct an IPP in which the verifier is given oracle access to a function $f : \mathbb{F}^m \to \mathbb{F}$ and needs to verify that f is close to a polynomial of individual degree $|H|-1$ that obtains only Boolean values in H^m such that their sum modulo 3, over the subcube H^m, is 0. The verifier may interact with the prover for $O(r^2)$ rounds.

As its initial step, our verifier checks that the given input f is close to some low degree polynomial by invoking the low degree test. This test, introduced by Rubinfeld and Sudan [59], ensures that if f is far from every low degree polynomial, then the verifier will reject with high probability. Thus, we can assume that f is close to some low degree polynomial. Moreover, using the self-correction property of polynomials, this means that with a small overhead, we can treat f as though it were itself a low degree polynomial (rather than just being close).[3]

Given this initial step, we can now assume without loss of generality that the function $f : \mathbb{F}^m \to \mathbb{F}$ is in fact a low degree polynomial. However, the verifier still needs to check that $\sum_{z \in H^m} f(z) = 0 \pmod 3$ and that $f|_{H^m} : H^m \to \{0,1\}$. For now though, let us focus

[2] We remark that a similar result could be obtained if we replaced the modulus 3 and the field's characteristic by any two *distinct* and *constant-sized* primes.

[3] Loosely speaking, the self-correction property of polynomials says that if f is guaranteed to be close to a low degree polynomial P, then one can read values from P by only making few queries to f. See Lemma 30 for the precise statement.

on the former task, which is the main step in our proof: checking that $\sum_{z \in H^m} f(z) = 0 \pmod 3$ (and we just assume that $f|_{H^m} : H^m \to \{0,1\}$).

Viewing $f|_{H^m}$ as a string $x \in \{0,1\}^k$, we need to construct an interactive proof in which the verifier uses oracle access to $\mathsf{LDE}(x)$ to verify that $\mathsf{wt}(x) = 0 \pmod 3$ in sublinear time. We refer to this type of proof-system, in which the verifier is given oracle access to an *encoding* of the input and runs in sublinear time, as a *holographic*[4] *interactive proof* (HIP).

More precisely, we say that a language has an HIP, with respect to some error-correcting code C, if it has an interactive proof in which the verifier has oracle access to an encoding under C of the input and verifies membership in the language using few queries to this encoding. The redundant representation of the input often allows the verifier to run in *sub-linear time.* We remark that HIPs play a central role in this work and we discuss them more in Section 1.3.

Thus, our task is now to construct an HIP (with respect to the LDE code) for the language

$$\mathcal{L}_{\mathsf{MOD3}} \overset{\text{def}}{=} \left\{ x \in \{0,1\}^k \ : \ \mathsf{wt}(x) = 0 \pmod 3 \right\}.$$

Before describing the construction of an HIP for $\mathcal{L}_{\mathsf{MOD3}}$, it will be instructive to consider as a warm-up, the construction of an HIP for the related language $\mathcal{L}_{\mathsf{MOD2}} = \{x \in \{0,1\}^k : \mathsf{wt}(x) = 0 \pmod 2\}$, where the important distinction is that the modulus 2 is also the characteristic of the field $\mathbb{F}$ under which x is encoded.

In this warmup case, we assume that the verifier is given oracle access to a polynomial $f : \mathbb{F}^m \to \mathbb{F}$ that obtains Boolean values in H^m (i.e. $f|_{H^m} : H^m \to \{0,1\}$), and needs to check that $\sum_{z \in H^m} f(z) = 0$, where the sum is over $\mathsf{GF}(2)$. Importantly, since we assumed that $f|_{H^m}$ is Boolean valued, and that the field $\mathbb{F}$ has characteristic 2, we can instead take the sum over the field $\mathbb{F}$ (rather than taking the integer sum mod 2).

The latter problem, of checking whether the sum of a given input polynomial is 0 over a subcube of its domain (i.e., over H^m), has a well-known interactive proof due to Lund *et al.* [52], which is often referred to as the *sumcheck protocol.* In this protocol the verifier only needs to query the polynomial f at a single point and so it can be viewed as an HIP. Furthermore, there are known variants of the sumcheck protocol that offer a suitable tradeoff between the number of rounds and verifier's complexity, which suffice for our purposes (i.e., an r-round IPP with verification time roughly $n^{1/r}$).

The aforementioned variants of the sumcheck protocol suffice for an upper bound for the warmup case. However, we do not know how to prove a corresponding lower bound, which is the reason that we set the modulus in our construction to be different from the field's characteristic.[5] While the original language $\mathcal{L}_{\mathsf{MOD3}}$ allows us to prove the desired lower bound, unfortunately it makes obtaining an upper bound more challenging. We proceed to the actual problem at hand: constructing an HIP (with respect to the LDE code over a field of characteristic 2) for checking $\mathcal{L}_{\mathsf{MOD3}}$.

Since the modulus and characteristic are different, our task can no longer be expressed as a linear constraint (over $\mathbb{F}$) on the bits of x. Since we do not know how to solve this problem directly using the sumcheck protocol, we turn to more complex interactive proofs from the literature. Specifically, our starting point will be the interactive proof-system of Goldwasser, Kalai and Rothblum [37], which we shall refer to as the GKR protocol.[6]

[4] The terminology of "holographic" interactive proofs originates from the "holographic proofs" of Babai *et al.* [5], which refers to probabilistic proof systems for *encoded* inputs. The notion of HIP, and its relation to other notions, is further discussed in Section 1.3.

[5] We conjecture that for the warmup case (i.e., when the modulus is 2) a lower bound that (roughly) corresponds to the upper bound given by the sumcheck protocol does hold.

[6] In Section A.1 we discuss our reason for basing our protocol on the GKR proof-system, rather than

The GKR Protocol. Goldwasser *et al.* give an interactive proof for any language computable by a logspace-uniform circuit of size S and depth D such that the number of rounds in their protocol is $D \cdot \mathsf{polylog}(S)$, the communication is also $D \cdot \mathsf{polylog}(S)$, and the verifier runs in time $(n + D) \cdot \mathsf{polylog}(S)$. Their protocol is based on algebraic techniques and, in particular, uses ideas originating from the interactive proof and PCP literature (cf., [63]). Our HIP for MOD3 will be based on a variant of their proof system.

Observe that one can check whether a given string x's Hamming weight is divisible by 3 using a highly uniform logarithmic-depth formula.[7] Thus, applying the GKR result gives us an interactive proof for $\mathcal{L}_{\mathsf{MOD3}}$. Most importantly for our purposes, if the GKR verifier is given oracle access to the LDE encoding of the input, then it only needs to check a single (random) element from the encoding (and in particular runs in sublinear time). In other words, the GKR protocol can be thought of as an HIP with *sublinear* time verification.[8] While the GKR protocol does yield an HIP for $\mathcal{L}_{\mathsf{MOD3}}$, its round complexity is poly-logarithmic and therefore too large for our purposes (recall that we are aiming for constant round protocols). The large round complexity is due to the fact that the high-level strategy in the GKR protocol is to process the circuit layer by layer, where the transition between each two consecutive layers uses an interactive protocol, which itself is based on the sumcheck protocol.

Even if we were to use a constant-round variant of the sumcheck protocol for each transition, the GKR protocol still uses $\Omega(D)$ rounds, where D is the depth of the circuit, which in our case is logarithmic and therefore too large. To get around this, we rely on an unpublished observation, due to Kalai and Rothblum [45], which shows that for every constant $r \geq 1$, if the circuit satisfies an extreme (and somewhat unnatural) uniformity condition[9], then $\log(n)/r$ layers can be processed at once, using r rounds of interaction and roughly $n^{1/r}$ communication. Thus, overall, a logarithmic depth circuit can processed in $O(r^2)$ rounds. Using this observation, [45] obtain *constant-round* interactive-proofs for all languages in NC^1 that satisfy the aforementioned uniformity condition.[10]

In our actual construction we do not use the [45] protocol directly (even though the language $\mathcal{L}_{\mathsf{MOD3}}$ satisfies the desired uniformity), but rather give a special purpose protocol tailored for $\mathcal{L}_{\mathsf{MOD3}}$ (which is inspired by their techniques). Doing so allows us to avoid stating their somewhat cumbersome uniformity condition and to introduce other simplifications (due to the simple and regular structure of the formula for $\mathcal{L}_{\mathsf{MOD3}}$). We proceed to describe this HIP.

other general purpose interactive proof-systems from the literature.

[7] E.g., consider the $\log(k)$-depth full binary tree with the input bits at its leaves, in which each internal vertex computes the sum modulo 3 of its two children, where each such modulo 3 sum can be computed by a simple constant size gadget composed of AND, OR and NOT gates.

[8] Note that obtaining an interactive-proof for $\mathcal{L}_{\mathsf{MOD3}}$ with a *linear-time* verifier is trivial, since the verifier can decide membership by itself in linear-time. The key benefit that we get from using the GKR protocol is that it allows for *sublinear* time verification given access to an encoded input.

[9] Loosely speaking, the uniformity condition requires that it be possible to compute *low degree extensions* of gate indicator functions that refer to gates of fan-in $t = n^{O(1/r)}$. That is, we view the formula as a depth r circuit consisting of gates of fan-in $t = n^{O(1/r)}$ (by grouping together every $\log(n)/r$ consecutive layers). For each of these r layers, and every type of fan-in t gate $g : \{0,1\}^t \to \{0,1\}$ that appears in that layer, we consider a gate indicator function I_g that given as input indices of $t+1$ wires, outputs 1 if the first wire is the result of an application of g to the other t wires. The [45] uniformity requirement is that it be possible to efficiently compute the *low degree extension* of I_g.

[10] Recall that the class NC^i consists of languages computable by polynomial-size $O\left((\log n)^i\right)$-depth circuits with fan-in 2. We emphasize that the [45] result gives *constant-round* protocols only for NC^1 circuits (that are sufficiently uniform), whereas the GKR result gives protocol with a *poly-logarithmic* round complexity for all (logspace uniform) languages in $\mathsf{NC} = \cup_{k \in \mathbb{N}} \mathsf{NC}^k$. (Furthermore, the GKR protocol for NC has poly-logarithmic communication complexity whereas the [45] protocol has $n^{1/O(1)}$ communication.)

A Holographic Interactive Proof for $\mathcal{L}_{\mathsf{MOD3}}$. Recall that we are given oracle access to a polynomial $X : \mathbb{F}^m \to \mathbb{F}$ promised to be the low-degree extension of a *Boolean* assignment $x \in \{0,1\}^k$, and our goal is to construct an $O(r^2)$-round HIP for verifying whether $x \in \mathcal{L}_{\mathsf{MOD3}}$. Also recall that we have fixed the parameters of the LDE code, including a field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$, and a dimension m such that $|H^m| = k$. However, for now we think of the sizes of these parameters as being $|H| = k^{1/r}$, $m = r$, and $|\mathbb{F}| = \mathsf{poly}(|H|, m)$, rather than $|H|$ being poly-logarithmic in k.[11]

For a given input polynomial $X : \mathbb{F}^m \to \mathbb{F}$ (of individual degree $|H| - 1$), we define a sequence of polynomials $V_0, \ldots, V_r$, where each $V_i : \mathbb{F}^i \to \mathbb{F}$ has individual degree $|H| - 1$ (note that these polynomials have gradually increasing domains). The polynomial $V_r : \mathbb{F}^r \to \mathbb{F}$ is defined as $V_r \equiv X$. The polynomials $V_1, \ldots, V_{r-1}$ are each defined to be the (unique) individual degree $|H| - 1$ polynomial that satisfies the following recursive relation:

$$\forall i \in [r],\ \forall h \in H^{i-1}, \quad V_{i-1}(h) = \sum_{\alpha \in H} V_i(h, \alpha) \pmod 3, \tag{1}$$

where the arithmetic is over the integers (modulo 3). Indeed, $V_0 \in \mathbb{F}$ is defined as a single field element $V_0 = \sum_{\alpha \in H} V_1(\alpha) \pmod 3$. Note that we identify the integers $\{0,1,2\}$ with three distinct elements in $\mathbb{F}$. Indeed, each of the V_i polynomials takes values in the set $\{0,1,2\} \subseteq \mathbb{F}$ over the subcube H^i.

Taking the [37, 45] view, each polynomial $V_i : \mathbb{F}^i \to \mathbb{F}$ can be thought of as the low degree extension of the i^{th}-layer (counting from the output layer) in a depth r formula of fan-in $k^{1/r}$ for $\mathcal{L}_{\mathsf{MOD3}}$ such that each gate computes the sum modulo 3 of its $k^{1/r}$ children. In particular,

$$V_0 = \sum_{\alpha \in H} V_1(\alpha) = \cdots = \sum_{h \in H^i} V_i(h) = \cdots = \sum_{h \in H^r} V_r(h) = \mathsf{wt}(x) \pmod 3.$$

Our main step is an interactive protocol that reduces a claim about an (arbitrary) single point in the polynomial V_{i-1} to a claim about a single (random) point in V_i. By applying this interactive reduction r times, we can reduce the initial claim $V_0 = 0$ to a claim about a single point in V_r, which we can explicitly check (since we have oracle access to $V_r \equiv X$). Each interactive reduction will take $O(r)$ rounds so overall we get an HIP for $\mathcal{L}_{\mathsf{MOD3}}$ with $O(r^2)$ rounds.

Towards showing such an interactive reduction protocol, we would like to express Equation (1), which is a modular equation over the integers, as a low degree relation over the field $\mathbb{F}$. Let $t \stackrel{\text{def}}{=} |H| = k^{1/r}$, and let $\xi_1, \ldots, \xi_t$ be the enumeration of all elements in H. Define the polynomial $\widetilde{\mathsf{MOD3}} : \mathbb{F}^t \to \mathbb{F}$ as the (unique) individual degree two polynomial such that for every $z \in \{0,1,2\}^t$, it holds that $\widetilde{\mathsf{MOD3}}(z) = \sum_{j \in [t]} z_i \pmod 3$, where the tilde in the notation is meant to remind us that $\widetilde{\mathsf{MOD3}}$ is not the modulo 3 summation function but rather its low degree extension over $\mathbb{F}$. Equation (1) can now be re-stated as:

$$\forall i \in [r],\ \forall h \in H^{i-1}, \quad V_{i-1}(h) = \widetilde{\mathsf{MOD3}}\Big(V_i(h, \xi_1), \ldots, V_i(h, \xi_t)\Big) \tag{2}$$

(where we use the fact that the V_i polynomials take values in $\{0,1,2\}$ over H^i.)

[11] We remark that setting $|H| = k^{1/r}$ is actually problematic for us since it induces a dependence between the language $\mathsf{Enc\text{-}MOD3}$ and the desired round complexity r. Nevertheless, it does yield a weaker hierarchy theorem in which we use a different language for each value of r. At the end of Section 1.2.1 we discuss how we overcome this difficulty.

Observe that Equation (2) is a polynomial relation between V_{i-1} and V_i that holds for inputs in H^{i-1}. We would like to obtain a similar relation for general inputs (i.e., in $\mathbb{F}^{i-1}$). To do so, we observe that, for every $z \in \mathbb{F}^{i-1}$, we can express $V_{i-1}(z)$ as an $\mathbb{F}$-linear combination of the values $\{V_{i-1}(h)\}_{h \in H^{i-1}}$ (this follows directly from the fact that the low degree extension is a *linear* code). We denote the coefficients in this linear combination by $\{\beta_z(h)\}_{h \in H^{i-1}}$ (these coefficients arise from Lagrange interpolation, but we ignore the specifics for this overview). Combining this observation together with Equation (2) we obtain:

$$\begin{aligned}\forall i \in [r],\ \forall z \in \mathbb{F}^{i-1}, \quad V_{i-1}(z) &= \sum_{h \in H^{i-1}} \beta_z(h) \cdot V_{i-1}(h) \\ &= \sum_{h \in H^{i-1}} \beta_z(h) \cdot \widetilde{\mathsf{MOD3}}\Big(V_i(h,\xi_1),\dots,V_i(h,\xi_t)\Big). \qquad (3)\end{aligned}$$

Using Equation (3) we will describe an interactive reduction from a claim about V_{i-1} to a claim about V_i. Suppose that our interactive reduction starts with a claim that $V_{i-1}(z_{i-1}) = \nu_{i-1}$ for some $z_{i-1} \in \mathbb{F}^{i-1}$ and $\nu_{i-1} \in \mathbb{F}$. By Equation (3) this translates into the claim:

$$\nu_{i-1} = \sum_{h \in H^{i-1}} \beta_{z_{i-1}}(h) \cdot \widetilde{\mathsf{MOD3}}\Big(V_i(h,\xi_1),\dots,V_i(h,\xi_t)\Big). \qquad (4)$$

We now observe that $Q_i(w) \overset{\text{def}}{=} \beta_{z_{i-1}}(w) \cdot \widetilde{\mathsf{MOD3}}\Big(V_i(w,\xi_1),\dots,V_i(w,\xi_t)\Big)$ is a low degree polynomial over $\mathbb{F}$ (since $\beta_{z_{i-1}}$, $\widetilde{\mathsf{MOD3}}$, and V_i have low degree). Thus, the claim in Equation (4) refers to the sum of a low degree polynomial over a subcube, which is precisely the problem that the sumcheck protocol solves.

It seems that we are done, except that a problem arises. In the sumcheck protocol the verifier is given oracle access to the polynomial whose sum over a subcube we wish to check. Although the polynomial Q_i on which we wish to run the sumcheck protocol is well-defined, our verifier does not have oracle access to it. Therefore it is not immediately clear how we can hope to run the sumcheck protocol with respect to Q_i.

We resolve this problem by noting that the sumcheck protocol can be used in an *input-oblivious* manner. In this variant, the verifier does not need to have oracle access to Q_i, but rather than accepting or rejecting, the verifier outputs a claim of the form $Q_i(w_{i-1}) = \gamma_{i-1}$, for some point $w_{i-1} \in \mathbb{F}^{i-1}$ and value $\gamma_{i-1} \in \mathbb{F}$. Completeness means that if the original claim is true (i.e., $\sum_{h \in H^{i-1}} Q_i(h) = \nu_{i-1}$), then the verifier always outputs (w_{i-1}, γ_{i-1}) such that $Q_i(w_{i-1}) = \gamma_{i-1}$, and soundness means that if the original claim is false (i.e., $\sum_{h \in H^{i-1}} Q_i(h) \neq \nu_{i-1}$), then for any cheating prover strategy, with high probability $Q_i(w_{i-1}) \neq \gamma_{i-1}$ (or the verifier rejects during the interaction). We stress that in this variant the verifier makes no queries to Q_i.[12] As for the number of rounds, recall that in the sumcheck protocol in each iteration one of the variables is "stripped" from the summation, which leads to a total of $i-1 \leq r$ rounds.

Having run the input-oblivious variant of the sumcheck protocool, our verifier is now left with the claim $Q_i(w_{i-1}) = \gamma_{i-1}$. However, to obtain our interactive reduction, we still need to reduce the foregoing claim to a claim about a (single) point in the polynomial V_i. To do so, the first idea that comes to mind is to have the prover provide the values

[12] To see that this variant is possible, observe that in the classical sumcheck protocol [52], the verifier only queries the polynomial at a single point and (at the end of the interaction) checks that it is equal to a particular value.

$\mu_j = V_i(w_{i-1}, \xi_j)$, for every $j \in [t]$. Given these values, the verifier can explicitly check that indeed $\gamma_{i-1} = \beta_{z_{i-1}}(w_{i-1}) \cdot \widetilde{\mathsf{MOD3}}(\mu_1, \dots, \mu_t)$.[13] If the prover indeed sent the correct values, then this last check assures us that indeed $Q_i(w_{i-1}) = \gamma_{i-1}$. However, since we cannot assume that the prover sent the correct values, we are left with t claim of the form $V_i(w_{i-1}, \xi_j) = \mu_j$, which the verifier needs to check.

Notice that we have actually reduced a single claim about V_{i-1} to t claims about V_i. This still falls short of our goal which was to reduce to only a *single* claim about V_i. (Indeed, we cannot afford to increase the number of claims by a t factor in each iteration, since this would yield a protocol with complexity $t^r = k$, which is trivial).

The final observation is that the points $\{(w_{i-1}, \alpha)\}_{\alpha \in H}$ lie on the (axis parallel) line $(w_{i-1}, *)$. Note that the restriction of a low degree polynomial to an axis parallel line is a low degree (univariate) polynomial. Thus, we will have the prover specify the entire polynomial $P_i : \mathbb{F} \to \mathbb{F}$ defined as $P_i(\alpha) = V_i(w_{i-1}, \alpha)$, for every $\alpha \in \mathbb{F}$. The verifier checks that $\gamma_{i-1} = \beta_{z_{i-1}}(w_{i-1}) \cdot \widetilde{\mathsf{MOD3}}\left(P_i(\xi_1), \dots, P_i(\xi_t)\right)$. The point is that now if the prover supplies an incorrect values for some $P_i(\alpha)$ (i.e., $P_i(\alpha) \neq V_i(w_{i-1}, \alpha)$), since both P_i and $V_i(w_{i-1}, *)$ are low degree polynomials, for most $\rho \in \mathbb{F}$ it holds that $P_i(\rho) \neq V_i(w, \rho)$. Thus, the verifier chooses at random $\rho_i \in \mathbb{F}$ and sets the claim for the next iteration to be $V_i(z_i) = \nu_i$, where $z_i = (w_{i-1}, \rho_i)$ and $\nu_i = P_i(\rho_i)$.[14]

To summarize, our HIP for $\mathcal{L}_{\mathsf{MOD3}}$ works in r phases. In the i^{th} phase we reduce a claim of the form $V_{i-1}(z_{i-1}) = \nu_{i-1}$, for some point $z_{i-1} \in \mathbb{F}^{i-1}$ and value $\nu_{i-1} \in \mathbb{F}$, into a claim $V_i(z_i) = \nu_i$, for $z_i \in \mathbb{F}^i$ and $\nu_i \in \mathbb{F}$ (which are generated during the interactive reduction). In particular, the first iteration begins with the claim $V_0 = 0$ (i.e., z_0 is the empty string and $\nu_0 = 0$), which corresponds to the claim that $x \in \mathcal{L}_{\mathsf{MOD3}}$ (i.e., $\mathsf{wt}(x) = 0 \pmod 3$). Thus, the i^{th} phase in our HIP begins with the claim $V_{i-1}(z_{i-1}) = \nu_{i-1}$. In the i^{th} phase, first the prover and verifier engage in the sumcheck protocol that arises from Equation (4). This yields the claim $Q_i(w_{i-1}) = \gamma_{i-1}$, for a point $w_{i-1} \in \mathbb{F}^{i-1}$ and value $\gamma_{i-1} \in \mathbb{F}$ (generated by the sumcheck protocol). Since the verifier has no access to Q_i, it asks the prover to send the polynomial $P_i : \mathbb{F} \to \mathbb{F}$ defined as $P_i(\alpha) = V_i(w_{i-1}, \alpha)$. The verifier checks that the values of this polynomial are consistent with the claim $Q_i(w_{i-1}) = \gamma_{i-1}$, and then selects a random point $\rho_i \in \mathbb{F}$. The claim for the following phase is that $V_i(z_i) = \nu_i$, where $z_i = (w_{i-1}, \rho_i)$ and $\nu_i = P_i(\rho_i)$. After r such phases we are left with the claim $V_r(z_r) = \nu_r$, for $z_r \in \mathbb{F}^r$ and $\nu_r \in \mathbb{F}$, which the verifier can explicitly check (since it has oracle access to $V_r \equiv X$).

The total number of rounds per interactive reduction is $O(r)$, and the communication complexity is roughly $\mathsf{poly}(t, r) = \mathsf{poly}(r, k^{1/r})$. Since we invoke r such reductions, overall we obtain an HIP for $\mathcal{L}_{\mathsf{MOD3}}$ with round complexity $O(r^2)$ and communication complexity $\mathsf{poly}(r, k^{1/r})$.

Obtaining an HIP over a Small Field. The approach outlined above yields an r^2-round HIP for $\mathcal{L}_{\mathsf{MOD3}}$, with respect to the code $\mathsf{LDE}_{\mathbb{F},H,m}$, in which the field size $|\mathbb{F}|$ is quite large (i.e., $|\mathbb{F}| \geq k^{1/r}$) and in particular depends on the value of r. Unfortunately, when we transform this HIP into an IPP for the language $\mathsf{Enc\text{-}MOD3}$, the dependence of the field size on r in the HIP introduces a dependence of the language $\mathsf{Enc\text{-}MOD3} \overset{\text{def}}{=} \{C(x) \;:\; x \in \{0,1\}^k \text{ with } \mathsf{wt}(x) = 0 \pmod 3\}$ on r. This dependence results in a weaker hierarchy theorem, in which we use a

[13] Note that both $\beta_{z_{i-1}}$ and $\widetilde{\mathsf{MOD3}}$ are *explicit* functions that the verifier can compute. Moreover they can even be computed *efficiently* using standard techniques, see the technical sections for details.

[14] We remark that this final step is actually very reminiscent of an individual round of the sumcheck protocol.

different language for each value of r . Our goal however is to obtain a *single* language, for which we can show an r-round IPP for every value of r (with a corresponding lower bound, which will be discussed in Section 1.2.2).

To this end we show a general reduction that transforms any HIP over a large field $\mathbb{F}$ into an HIP over a much smaller field $\mathbb{F}'$, as long as $\mathbb{F}$ is an extension field of $\mathbb{F}'$. We do so by showing that any $\mathbb{F}$-linear claim regarding the input (e.g., a claim about a single point in the $\mathsf{LDE}_{\mathbb{F},H,m}$ encoding) can be broken down (coordinate-wise) into d claims that are $\mathbb{F}'$-linear, where $d = \log(|\mathbb{F}|/|\mathbb{F}'|)$ is the degree of the field extension (i.e., $(\mathbb{F}')^d$ is isomorphic to $\mathbb{F}$). We can then easily verify each one of these $\mathbb{F}'$-linear claims using the sumcheck protocol over the smaller field $\mathbb{F}'$.[15]

We remark that the ability to switch fields when using (holographic) interactive proofs seems like a useful tool, and we believe that it will be useful in other contexts as well.

Checking Booleanity. In the above analysis we assumed for simplicity that the input $x = f|_{H^m}$ is Boolean valued. In order to actually check this, we follow an idea of Kalai and Raz [46] (which was used in the context of constructing interactive PCPs). We observe that the polynomial $f : \mathbb{F}^m \to \mathbb{F}$ is Boolean valued in a subcube H^m if and only if the (slightly higher degree) polynomial $g : \mathbb{F}^m \to \mathbb{F}$, defined as $g(z) = f(z) \cdot (1 - f(z))$ is identically 0 in H^m. The latter problem (of checking whether a polynomial vanishes on a particular subcube) can be solved via a relatively simple reduction to the sumcheck protocol, that has been used in the construction of PCPs.[16] We note that we crucially use fact that the reduction from f to g is local (i.e., the value of g at a point depends on the value of f at $O(1)$ points), and therefore can be used in our setting.

1.2.2 Lower Bound

We need to show a lower bound on the complexity of r-round IPPs for our language $\mathsf{Enc\text{-}MOD3} = \{C(x) \ : \ x \in \{0,1\}^k \text{ with } \mathsf{wt}(x) = 0 \pmod 3\}$, where $C : \mathbb{F}^k \to \mathbb{F}^n$ is the low degree extension code. Our lower bound will strongly use the fact that any $\mathbb{F}$-linear code (and in particular the low degree extension code that we use), for a field $\mathbb{F}$ of characteristic 2, is also a $\mathsf{GF}(2)$-linear code.

Our lower bound relies on a connection between IPPs and low-depth circuits, which was discovered by Rothblum, Vadhan and Wigderson [58]. Following their approach, in Section 4.3 we show that to prove an IPP lower bound for Enc-MOD3, it suffices to construct two distributions D_0 and D_1 over n-bit strings such that:

1. D_0 is distributed over the support of Enc-MOD3 (with high probability);
2. D_1 is far from Enc-MOD3 (with high probability); and
3. Every sufficiently small DNF formula cannot distinguish between inputs from D_0 and D_1 (with more than, say, 0.1 advantage).

[15] To obtain the desired *computational* efficiency for the latter task, we actually use the [45] protocol, which introduces an additional $O(r^2)$ rounds. We believe this use is an overkill, and we hope to replace it with a more elementary argument in a following revision.

[16] In a nutshell, to check whether $g|_{H^m} \equiv 0$ we consider the restriction of g to the domain H^m and take the low degree extension $\hat{g}$ of that partial function. We observe that g is identically 0 in H^m if and only if $\hat{g}$ is identically 0 in $\mathbb{F}^m$. Thus, it suffices to check whether for a random point $z \in \mathbb{F}^m$, which the verifier chooses, it holds that $\hat{g}(z) = 0$. The linearity of the LDE code now means that this check can be solved by invoking the sumcheck protocol. See Section 3.4 for details.

The two distributions that we consider are D_0 and D_1 such that D_b is uniform over the set $\{C(x) \,:\, x \in \{0,1\}^k \text{ and } \mathsf{wt}(x) = b \pmod 3\}$. Note that D_0 is the uniform distribution over $\mathsf{Enc\text{-}MOD3}$, and so satisfies requirement (1), whereas the fact that D_1 satisfies requirement (2) follows from the distance of the code C. To show that the third requirement holds, consider a DNF ϕ that distinguishes between D_0 and D_1. We show that the size of ϕ must be large. Consider the distributions D'_0 and D'_1 over k-bit strings defined as

$$D'_b = \{x \in \{0,1\}^k \,:\, \mathsf{wt}(x) = b \pmod 3\}.$$

We can easily construct from ϕ a circuit ϕ' that distinguishes between D'_0 and D'_1: the circuit ϕ' first computes the encoding $C(x)$ of its input $x \in \{0,1\}^k$, and then applies the DNF ϕ to the result. Using the fact that C is linear over $\mathsf{GF}(2)$, it follows that ϕ' is a DNF of parities (i.e., a depth-3 formula with an OR gate at the top layer, AND gates at the middle layer, and XOR gates at the bottom layer). Now, we can apply the Razborov-Smolensky [59] lower bound, which shows that any small $\mathsf{AC}_0[2]$ circuit (i.e., circuits of constant-depth circuits with AND, OR, and PARITY gates of unbounded fan-in), and in particular a DNF of parities, cannot even approximate the summation modulo 3 function (i.e., distinguish between D'_0 and D'_1).

1.3 Holographic Interactive Proofs

The proof of our hierarchy theorem utilizes a special type of interactive proofs, which we call *holographic interactive proofs*. A holographic interactive proof (HIP) is an interactive proof in which, instead of getting its input x explicitly, the verifier is given *oracle* access to $C(x)$, an error-corrected encoding of the input x, for a bounded number of queries. Hence, HIPs may be thought of as interactive proofs for promise problems of the form $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ with $\Pi_{\mathsf{YES}} = \{C(x) : x \in \mathcal{L}\}$ and $\Pi_{\mathsf{NO}} = \{C(x) : x \notin \mathcal{L}\}$.

The notion of HIP was used, either implicitly or explicitly as a technical tool that underlies many probabilistic proof systems (e.g., [52, 6, 5, 46, 37, 47, 58, 43, 48, 56, 28]).[17] These works demonstrate that, by using the redundant encoding of the input, we can often achieve sublinear verification time. (As a matter of fact, in most of these works, it suffices for the verifier to read just a *single* point in the encoding.) We remark that throughout this work (as well as in most previous works[18]), the specific code that is used is the *low-degree extension code* (LDE).

Some of the techniques that were outlined in Section 1.2.1, can be viewed as generic transformations on HIPs (with respect to the LDE code), and we present them as such in the technical parts of this work. These techniques include the ability to switch fields, or check Booleanity, and the connection to IPPs. We wish to highlight the conceptual importance of HIPs, and advocate a continued systematic study of these proof systems.

We also remark that HIPs with respect to the LDE code are closely related to interactive proofs in the algebrization framework [1]. In both models the verifier is given oracle access to a low degree polynomial and may interact with the prover to decide on some property of the "message" or "oracle" encoded within the polynomial. See Section 5 for further discussion of this connection.

[17] The first explicit use is in [5].

[18] A notable exception is the work of Meir [53], which is based on general tensor codes. We remark that using Meir's techniques it may be possible to extend our results to other tensor codes. We leave exploring this possibility to future work.

1.4 Related Works

In this section, we discuss several lines of works that are related to our work.

Interactive Proofs of Proximity. The notion of interactive proofs of proximity (IPP) was first considered by Ergün, Kumar and Rubinfeld [19]. Its study was re-initiated by Rothblum, Vadhan and Wigderson [58], who showed that every language computable by a low-depth circuit has an IPP with a sublinear time verifier. IPPs were further studied by [31, 28] who showed more efficient IPPs for certain restricted complexity classes. Other works have focusing on variants such as non-interactive (MA) proofs of proximity [43, 20, 30] and interactive *arguments* of proximity [49]. Proofs of proximity have also found applications to property testing and related models [33, 34, 21].

Hierarchy Theorems for Standard Interactive Proofs. Aiello, Goldwasser and Håstad [2] showed a round hierarchy theorem in a relativized world (i.e., with respect to an oracle). However, the later results of [52, 60], which are based on non-relativizing techniques, demonstrate that relativization is not an actual barrier, especially in the context of interactive proofs.[19] We note that although they are technically quite different, both our lower bound and the lower bound of [2] are based on circuit lower bounds for low depth circuits.

Goldreich, Vadhan and Wigderson [36] showed a *conditional* round hierarchy result for standard interactive proofs, based on the assumption that co-SAT does not have a 1-round $\mathcal{AM}$ proof-system with complexity $2^{o(n)}$.[20] We emphasize that the result of [36] is based on an unproven and arguably strong (yet believable) assumption, whereas our result is unconditional.

We also note that for *computationally sound* proofs, also known as arguments, under reasonable cryptographic assumptions there are extremely efficient 2-round protocols [50] and even 1-round protocols [48]. In particular, these results show that the power of arguments does not scale with additional rounds (since a fixed constant number of rounds suffice). A similar statement holds for arguments of proximity that are the computationally sound variant of IPPs (see [58, 49]).

Interactive PCPs. Holographic interactive proofs (HIPs) are closely related to the notion of *interactive* PCPs, introduced by Kalai and Raz [46]. Roughly speaking, interactive-PCPs are encodings of NP-witnesses that, like PCPs can be verified using few queries, but here the verification procedure may use interaction with an unbounded (and untrusted) prover. Thus, using our terminology, an interactive PCP can be thought of as an HIP for checking the NP witness relation.

Arthur-Merlin Query Complexity. Every IPP for a language $\mathcal{L}$ can be viewed as a protocol, for a promise problem related to $\mathcal{L}$, in the Arthur Merlin query complexity model, previously studied by Raz *et al.* [54]. This model, similarly to IPPs, considers a sub-linear time verifier, that is given oracle access to an input and may interact with an (untrusted) prover. Indeed, one may view IPPs as Arthur Merlin query complexity protocols which focus on promise problems in which the goal of the verifier is to distinguish between inputs having a certain property from those that are *far* from having the property.

[19] Indeed, Fortnow and Sipser [22] show that the proof of IP = PSPACE cannot be relativized (in fact, IP does not even contain coNP relative to a random oracle [14]). In fact, the algebrization framework of Aaronson and Wigderson [1] was proposed precisely to address this issue. Connections between our results and algebrization are further discussed in Section 5.

[20] Related assumptions have recently been studied also by Carmosino *et al.* [10] and Williams [69].

Thus, our main result directly yields a round hierarchy theorem (for a promise problem) in the *Arthur-Merlin Query Complexity* model and a sub-exponential separation between the complexity of constant-round vs. general (i.e., unbounded round) Arthur-Merlin Query Complexity protocols.

Interactive Proofs in Other Models. Interactive proof systems were studied also in the communication complexity setting (e.g., [7, 51, 61, 41, 40]). Here Alice and Bob may interact with an untrusted Merlin, who sees both of their inputs. We remark that showing any non-trivial explicit lower bound in the $\mathcal{AM}$ variant of this model, much less a hierarchy of separations, is a notorious open problem.

A recent line of works has studied interactive proofs in the data streaming model (e.g., [12, 16, 17, 42, 11, 65, 18]). Most relevant is a result of Chakrabarti *et al.* [13], who show a hierarchy theorem for the first four levels in the model of *online interactive proofs* (with exponential separations between these four levels).

Universal Locally Verifiable Codes. In a recent work, Goldreich and Gur [29] introduced the notion of *universal locally verifiable codes* (universal-LVC), which is closely related to holographic interactive proofs. A universal-LVC $C : \{0,1\}^k \to \{0,1\}^n$ for a family of functions $\mathcal{F} = \left\{f_i : \{0,1\}^k \to \{0,1\}\right\}_{i\in[M]}$ is a code such that for every $i \in [M]$, membership in the subcode $\{C(x) \ : \ f_i(x) = 1\}$ can be verified locally given an explicit access to a short (sublinear length) proof; put differently, for every $i \in [M]$ there exists a 1-message IPP for the property $\{C(x) \ : \ f_i(x) = 1\}$, with sublinear communication and query complexity.

1.5 Organization

In Section 2 we define IPPs and introduce some notations and definitions that we use throughout this work. In Section 3 we define holographic interactive proofs (HIPs) and prove some general results on them. In Section 4, using some of the results of Section 3, we prove the hierarchy theorem. Lastly, in Section 5 we discuss the implications to classical complexity theory.

Some of the discussion and proofs are deferred to the appendix. In Appendix A.2 we discuss an alternative language for the round hierarchy theorem and our choice of basing our protocol on GKR rather than, say a recent protocol of Reingold *et al.* [56]. Appendices B to D contain some standard proofs that are included for completeness.

2 Preliminaries

We begin with some standard notations:

- We denote the relative distance, over alphabet Σ, between two strings $x \in \Sigma^n$ and $y \in \Sigma^n$ by $\Delta(x,y) \overset{\text{def}}{=} \frac{|\{x_i \neq y_i \ : \ i\in[n]\}|}{n}$. If $\Delta(x,y) \le \varepsilon$, we say that x is ε-close to y, and otherwise we say that x is ε-far from y. Similarly, we denote the relative distance of x from a non-empty set $S \subseteq \Sigma^n$ by $\Delta(x,S) \overset{\text{def}}{=} \min_{y\in S} \Delta(x,y)$. If $\Delta(x,S) \le \varepsilon$, we say that x is ε-close to S, and otherwise we say that x is ε-far from S.
- We denote the projection of $x \in \Sigma^n$ to a subset of coordinates $I \subseteq [n]$ by $x|_I$ and, for $i \in [n]$, write $x_i = x|_{\{I\}}$ to denote the projection to a singleton.

An additional notation that we will use is that if $S = (S_k)_{k\in\mathbb{N}}$ and $T = (T_k)_{k\in\mathbb{N}}$ are ensembles of sets, we denote by $S \subseteq T$ the fact that $S_k \subseteq T_k$ for every $k \in \mathbb{N}$.

Integrality. Throughout this work, for simplicity of notation, we use the convention that all (relevant) integer parameters that are stated as real numbers are implicitly rounded to the closest integer.

2.1 Interactive Proofs of Proximity

A language is an ensemble $\mathcal{L} = (\mathcal{L}_n)_{n\in\mathbb{N}}$, where $\mathcal{L}_n \subseteq (\Sigma_n)^n$ for every $n \in \mathbb{N}$ and where $\Sigma = (\Sigma_n)_{n\in\mathbb{N}}$ is the alphabet.

▶ **Definition 3** (Interactive Proofs of Proximity (IPP)). Let $\Sigma = (\Sigma_n)_{n\in\mathbb{N}}$ be an alphabet ensemble. An r-round *interactive proof of proximity*, with respect to proximity parameter $\varepsilon > 0$, (in short, ε-IPP) for the language $\mathcal{L}$ is an interactive protocol between a prover $\mathcal{P}$, which gets *free* access to ε and to an input $x \in \Sigma^n$, and a verifier $\mathcal{V}$, which gets free access only to ε and n, as well as *oracle* access to x. At the end of the protocol, the following conditions are satisfied:

- **Completeness**: If $x \in \mathcal{L}$, then, when $\mathcal{V}$ interacts with $\mathcal{P}$, with probability $2/3$ it accepts.
- **Soundness**: If x is ε-far from $\mathcal{L}$, then for every prover strategy $\mathcal{P}^*$, when $\mathcal{V}$ interacts with $\mathcal{P}^*$, with probability $2/3$ it rejects.

If the completeness condition in Definition 3 holds with probability 1, then we say that the IPP has perfect completeness. A public-coin IPP is an IPP in which every message from the verifier to the prover consists only of fresh random coin tosses.

An IPP is said to have query complexity $q : \mathbb{N} \times [0, 1] \to \mathbb{N}$ if for every $n \in \mathbb{N}$, $\varepsilon > 0$, $x \in \{0, 1\}^n$, and any prover strategy $\mathcal{P}^*$, the verifier makes at most $q(n, \varepsilon)$ queries to x when interacting with $\mathcal{P}^*$. The IPP is said to have communication complexity $c : \mathbb{N} \times [0, 1] \to \mathbb{N}$ if for every $n \in \mathbb{N}$, $\varepsilon > 0$, and $x \in \mathcal{L}_n$ the communication between $\mathcal{V}$ and $\mathcal{P}$ consists of at most $c(n, \varepsilon)$ bits.

2.2 Constructible Error Correcting Codes and Finite Fields

An error correcting code over an alphabet Σ is an injective function $C : \Sigma^k \to \Sigma^n$. The code C is said to have relative distance δ if for any $x \neq x' \in \Sigma^k$ it holds that $\Delta(x, x') \geq \delta$.

Throughout this work we deal with (uniform) polynomial-time algorithms, and so we will need (families of) codes that are efficiently computable. Formally, for a parameter $n = n(k) \geq 1$ that is called the blocklength, and ensemble of alphabets $\Sigma = (\Sigma_k)_{k\in\mathbb{N}}$, we define a constructible error correcting code over Σ as an ensemble $C = \left(C_k : \Sigma_k^k \to \Sigma_k^n\right)_{k\in\mathbb{N}}$ of error correcting codes, such that the function $f(x) = C_{|x|}(x)$ is computable by a polynomial-time Turing machine (in particular this implies that $n = \mathsf{poly}(k, \log(\Sigma))$). An ensemble of error correcting codes $C = (C_k)_{k\in\mathbb{N}}$ is said to have relative distance δ if for all sufficiently large k, each code C_k in the ensemble has relative distance δ.

Throughout this work, we mostly consider codes defined over finite fields (i.e., the alphabets Σ_k are all finite fields). Such codes are called linear if they are linear functions over the field.

Finite Fields and Polynomials. Many of our algorithms and interactive proofs deal with finite fields. We consider ensembles of finite fields $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$, where $|\mathbb{F}_k|$ and say that such ensembles are constructible if the field operations can be done in $\mathsf{poly}\log(|\mathbb{F}_k|)$ time. Namely, there exist a Turing machine that given as input k and an appropriate number of elements in $\mathbb{F}_k$ (represented as strings of length $O(\log(|\mathbb{F}_k|))$ bits) can compute the field operations (i.e., addition, subtraction, multiplication, inversion, and sampling random elements) in $\mathsf{polylog}(|\mathbb{F}_k|)$ time.

The following fact shows that there exist constructible finite fields of characteristic 2.

▶ **Fact 4.** *For every time-constructible function $f = f(k) \geq 1$, there exists a* constructible *field ensemble $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ such that $|\mathbb{F}| = O(f)$ and $\mathbb{F}_k$ has characteristic 2 (i.e., is an extension field of* $\mathsf{GF}(2)$*) for every $k \in \mathbb{N}$.*

For details see [24, Appendix G.3] and references therein. We will also use the well-known Schwartz-Zippel Lemma.

▶ **Lemma 5** (Schwartz-Zippel Lemma)**.** *Let $P : \mathbb{F}^m \to \mathbb{F}$ be a non-zero polynomial of total degree d over the field $\mathbb{F}$. Then,*

$$\Pr_{x\in_R\mathbb{F}^m}[P(x) = 0] \leq \frac{d}{|\mathbb{F}|}.$$

2.3 Low-Degree Extension

Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be an ensemble of fields, and let $H = (H_k)_{k\in\mathbb{N}} \subseteq \mathbb{F}$ (the notation $H \subseteq \mathbb{F}$ means that $H_k \subseteq \mathbb{F}_k$, for every $k \in \mathbb{N}$). Let $m = m(k) \geq 1$ be a parameter, which we often call the dimension.

A basic fact is that for every function $f : H^m \to \mathbb{F}$ there exists a *unique* function $\tilde{f} : \mathbb{F}^m \to \mathbb{F}$ such that $\tilde{f}$ is a polynomial with individual degree $|H| - 1$ that agrees with f on H^m. Moreover, there exists an individual degree $|H| - 1$ polynomial $\beta : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ such that for every function $f : H^m \to \mathbb{F}$ it holds that

$$\tilde{f}(z) = \sum_{x\in H^m} \beta(x, z) \cdot f(x).$$

The function $\tilde{f}$ is called the low degree extension of $\mathbb{F}$ (with respect to the field $\mathbb{F}$, subset H and dimension m).

The following two propositions show that the low degree extension encoding can be computed efficiently.

▶ **Proposition 6.** *Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be a constructible field ensemble, let $H = (H_k)_{k\in\mathbb{N}} \subseteq \mathbb{F}$ be an ensemble of subsets and let $m = m(k)$ be the dimension.*

There exists a Turing machine that on input k runs in time $\mathsf{poly}(|H|, m, \log|\mathbb{F}|)$ and space $O(\log(|\mathbb{F}|) + \log(m))$, and outputs the polynomial $\beta : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ defined above, represented as an arithmetic circuit over $\mathbb{F}$.

Moreover, the arithmetic circuit β can be evaluated in time $\mathsf{poly}(|H|, m, \log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|) + \log(m))$. Namely, there exists a Turing machine with the above time and space bounds that given an input pair $(x, z) \in \mathbb{F}^m \times \mathbb{F}^m$ outputs $\beta(x, z)$.

See, e.g., [57, Proposition 3.2.1] for a proof of Proposition 6.

▶ **Proposition 7.** *Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be a constructible field ensemble, let $H = (H_k)_{k\in\mathbb{N}} \subseteq \mathbb{F}$ be an ensemble of subsets and let $m = m(k)$ be the dimension.*

Let $\phi : H^m \to \mathbb{F}$ and suppose that ϕ can be evaluated by a Turing Machine in time t and space s. Then, there exists a Turing machine that, given as an input a point $z \in \mathbb{F}^m$, runs in time $|H|^m \cdot (\mathsf{poly}(|H|, m, \log(|\mathbb{F}|)) + O(t))$ and space $O(m \cdot \log(|H|) + s + \log(|\mathbb{F}|))$ and outputs the value $\hat{\phi}(z)$ where $\hat{\phi}$ is the unique low degree extension of ϕ (with respect to $H, \mathbb{F}, m$).

Proof. The Turing machine computes

$$\hat{\phi}(z) = \sum_{x\in H^m} \beta(x, z) \cdot \phi(x)$$

by generating and evaluating β as in Proposition 6. ◀

Low Degree Extension as an Error-Correcting Code. The low degree extension can also be viewed as an error-correcting code in the following way. Suppose that H and m are such that $|H|^m = k$. Then, we can associate a string $x \in \mathbb{F}^k$ with a function $x : H^m \to \mathbb{F}$ by identifying H^m with $[k]$ in some canonical way.

We define the low degree extension of a string x as $\mathsf{LDE}_{\mathbb{F},H,m}(x) = \tilde{x}$. That is, the function $\mathsf{LDE}_{\mathbb{F},H,m}$ is given as input the string $x \in \mathbb{F}^k$, views it as a function $x : H^m \to \mathbb{F}$ and outputs its low degree extension $\tilde{x}$. By Proposition 7 the code $\mathsf{LDE}_{\mathbb{F},H,m}$ is constructible, and by the Schwartz-Zippel Lemma (Lemma 5), the code $\mathsf{LDE}_{\mathbb{F},H,m}$ has relative distance $1 - \frac{m \cdot |H|}{|\mathbb{F}|}$.

3 Holographic Interactive Proofs

In this section we define *holographic interactive proofs* and show several transformations and generic results (which will be used in Section 4 for the proof of the hierarchy theorem). In Section 3.1 we give a formal definition and some basic facts. Having read Section 3.1, the reader may freely skip the rest of Section 3 and proceed directly to Section 4, which is the main technical section, and return to read the results of Sections 3.2 to 3.4 when they are used in Section 4.

Sections 3.2 to 3.4 focus on HIPs with respect to the low degree extension encoding. In Section 3.2 we show that such HIPs imply interactive proofs of *proximity* (for a related language). In Section 3.3 we show that one can switch the field under which the HIPs input is encoded (at a moderate cost) to any other field *that shares the same characteristic.* Finally, in Section 3.4 we show that HIPs can efficiently verify that the input (which can presumably be an arbitrary vector over the field) is actually Boolean valued (i.e., in $\{0,1\}^k$).

3.1 Definition and Basic Facts

A *holographic interactive proof* is similar to a standard interactive proof, except that rather than getting the input explicitly, the verifier gets oracle access to an encoding of the input (via an error correcting code). Using this redundant representation, we could potentially hope to have protocols in which the verifier runs in *sublinear* time and, in particular, does not even read its entire input. This hope is indeed materialized in several protocols from the literature (e.g., [52, 37, 56]).

As a matter of fact, it turns out that for some codes (specifically the low degree extension), reading just a single point p from the encoded input suffices for the verifier.[21] Thus, we restrict our attention to such protocols. Furthermore, in order to facilitate composition, rather than having the verifier actually read the (encoded) input at the point p, the verifier outputs a claim about the point (i.e., it outputs p together with a symbol that it would have expected to see, had it actually queried the (encoded) input at p).

Formally, holographic interactive proofs are parametrized by a (constructible) error correcting code C, under which the input is encoded, and are defined as follows.

▶ **Definition 8** (Holographic Interactive Proofs (HIP))**.** Let $\Sigma = (\Sigma_k)_{k\in\mathbb{N}}$ and $\Lambda = (\Lambda_k)_{k\in\mathbb{N}}$ be alphabet ensembles such that $\Lambda \subseteq \Sigma$. Let $\mathcal{L} \subseteq \Lambda$, and let $C : \Sigma^k \to \Sigma^n$ be a constructible error correcting code.

[21] For the low degree extension this can be shown to hold generically. The high level idea is to consider a low degree curve passing through all the points that the verifier wishes to read. The prover specifies the values for all the points on the curve and the verifier checks the provided answer on a random point on the curve. Soundness follows from the fact that composing a low-degree curve with a low-degree polynomial results in a low degree univariate polynomial. See, e.g., [46, Section 6] for details.

An r-round public-coin *holographic interactive proof (*HIP*)* for the language $\mathcal{L}$, with respect to the code C, is an interactive protocol between a prover $\mathcal{P}$, which gets as input $x \in \Sigma^k$, and a verifier $\mathcal{V}$, which gets as input only k. At the end of the protocol either the verifier rejects or it outputs a coordinate $i \in [n]$ and a symbol $\sigma \in \Sigma$ such that:

- **Completeness**: If $x \in \mathcal{L}$, then, when $\mathcal{V}$ interacts with $\mathcal{P}$, with probability 1 it outputs (i, σ) such that $C(x)|_i = \sigma$.
- **Soundness**: If $x \notin \mathcal{L}$, then for every prover strategy $\mathcal{P}^*$, when $\mathcal{V}$ interacts with $\mathcal{P}^*$, with probability $1 - \varepsilon$ either $\mathcal{V}$ rejects or it outputs (i, σ) such that $C(x)|_i \neq \sigma$, where $\varepsilon = \varepsilon(k) \in [0, 1]$ is called the *soundness error*.

In this work, all the holographic proofs that we consider are with respect to the low degree extension code (using a variety of different parameters), which was defined in Section 2.3 above.

▶ Remark (Different Alphabets for the Language and the Code). Typically, when using HIPs the alphabet Λ over which the language is defined will be the same as the alphabet Σ over which the code is defined. Still, in some cases it will be convenient for us to present HIPs that only work for particular sub-alphabets of the code (e.g., when the input is binary but the code is more naturally defined over some large alphabet) and so we give this more flexible definition.

Our definition of HIPs tries to capture many of the known interactive proof-systems in the literature, while being flexible and easy to compose. Indeed, the fact that HIPs can be transformed into standard interactive proofs which is immediate, is captured by the following proposition.

▶ **Proposition 9.** *Let $\Sigma = (\Sigma_k)_{k \in \mathbb{N}}$ and $\Lambda = (\Lambda_k)_{k \in \mathbb{N}}$ be alphabets such that $\Lambda \subseteq \Sigma$. Let $\mathcal{L}$ be a language over the alphabet Λ and let $C : \Sigma^k \to \Sigma^n$ be a constructible error correcting code.*

Any HIP *for $\mathcal{L}$ can be converted into a standard interactive proof with only a* $\mathsf{poly}(n)$ *additive overhead to the verifier's running time (and all other parameters remain unchanged). Moreover, the precise overhead is equal to the time that it takes to compute the i^{th} character of $C(x)$, given $x \in \Lambda^k$ and the index $i \in [n]$.*

Proof. The prover and verifier run the HIP. If the HIP verifier rejects, then we immediately reject. Otherwise, the HIP verifier outputs a pair $(i, \sigma) \in [n] \times \Sigma$ with the associated claim $C(x)|_i = \sigma$. We can now check this claim directly by computing $C(x)|_i$ and comparing with σ. ◀

The Sumcheck Protocol (as an HIP). We will make extensive use of the classical sumcheck protocol of Lund *et al.* [52]. Recall that the sumcheck protocol is an interactive proof for verifying that the sum, over a subcube, of a low degree polynomial is zero. Our protocol differs slightly from the "textbook" sumcheck protocol in two ways:

1. The verifier does not actually read any points from the input polynomial. Rather, at the end of the protocol it outputs a claim about a single point of the polynomial (i.e., the protocol is an HIP).
2. Following other works in the literature, our protocol allows a trade-off between the number of rounds and the communication complexity (rather than having the number of rounds correspond exactly to the dimension of the polynomial).

▶ **Lemma 10** (Sumcheck as an HIP). *Let $\mathbb{F}$ be a constructible field ensemble and let $H \subseteq \mathbb{F}$ be an ensemble of subsets of $\mathbb{F}$. Let $m = m(k)$ be an ensemble of integers such that $m = \log_{|H|}(k)$.*

Let $\mathcal{L} = \cup_{k\in\mathbb{N}}\mathcal{L}_k$, where $\mathcal{L}_k = \{x \in \mathbb{F}^k : \sum_{i\in[k]} x_i = 0\}$ and where the summation is over the field $\mathbb{F}$. Then, for every $r \in [m]$, there exists an r-round (public-coin) HIP *for $\mathcal{L}$, with respect to the code* $\mathsf{LDE}_{\mathbb{F},H,m}$*, with soundness error $\frac{m\cdot|H|}{|\mathbb{F}|}$ and communication complexity $|H|^{\lceil m/r\rceil} \cdot r \cdot \log|\mathbb{F}|$. The verifier runs in time $|H|^{\lceil m/r\rceil} \cdot r \cdot \mathsf{polylog}(|\mathbb{F}|)$ and the prover runs in time* $\mathsf{poly}(|\mathbb{F}|^m, r)$.

The proof of Lemma 10, which is standard, is included for completeness in Appendix C.

3.2 From HIP to IPPs

Proposition 9 above, shows that an HIP can be easily transformed into a standard interactive proof. We now show that HIPs, with respect to the low degree extension encoding, can be easily transformed into highly efficient (and in particular sublinear) *interactive proof of proximity* (IPP) for a related language. More specifically, we transform an HIP for the language $\mathcal{L}$ with respect to the $\mathsf{LDE}_{\mathbb{F},H,m}$ code, into an IPP for the language $\mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L}) \stackrel{\text{def}}{=} \{\mathsf{LDE}_{\mathbb{F},H,m}(x) : x \in \mathcal{L}\}$.[22]

► **Lemma 11.** *Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be an ensemble of finite fields, let $H = (H_k)_{k\in\mathbb{N}}$ be an ensemble of subsets (i.e. $H \subseteq \mathbb{F}$) and let $m = m(k)$ be such that $|H|^m = k$.*

Suppose that the language $\mathcal{L}$ has an r-round HIP*, with respect to the code* $\mathsf{LDE}_{\mathbb{F},H,m}$*, with communication complexity c. Then, the language* $\mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L})$ *has an r-round ε-*IPP *with query complexity $O(|H| \cdot m \cdot 1/\varepsilon)$ and communication complexity c.*

The key observations that we use to prove Proposition 11 are that (1) the IPP verifier can first check that its input is close to a low degree polynomial using low degree test. If the test passes, then, using the self-correctability of polynomials, the IPP verifier can emulate access to the encoded input of the HIP. Given these two observations the proof of Proposition 11 is standard and so we defer it to Appendix B.

3.3 Field Switching

In this subsection we show that HIPs can evaluate points in a LDE over an *extension* field of the base field under which the input is actually encoded. This fact is used in the proof Lemma 17 and allows us to first construct an HIP over a large field, and later convert it into an HIP over the smaller field.

The key observation for our field switching, is that verifying a linear claim involving the LDE over an extension field $\mathbb{K}/\mathbb{F}$ can be reduced to verifying several linear claims over the base field $\mathbb{F}$. Each of these linear claims can be verified via a sumcheck protocol (in fact, it suffices to verify a random linear combination of these claims), and so an HIP can emulate access to the LDE over the extension field $\mathbb{K}$ by making queries to the LDE over field $\mathbb{F}$. We proceed to the formal statement and proof.

Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ and $\mathbb{K} = (\mathbb{K}_k)_{k\in\mathbb{N}}$ be constructible field ensembles such that $\mathbb{K}$ is a degree $s = s(k) \le \log(k)$ field extension of $\mathbb{F}$ (i.e., $\mathbb{K}_k \cong \mathbb{F}_k^{s(k)}$, for every $k \in \mathbb{N}$). Let $H = (H_k)_{k\in\mathbb{N}} \subseteq \mathbb{F}$ and $G = (G_k)_{k\in\mathbb{N}} \subseteq \mathbb{K}$ be ensembles of subsets of $\mathbb{F}$ and $\mathbb{K}$, respectively. Let $m = m(k)$ and $\ell = \ell(k)$ be ensembles of integers such that $|H|^m = |G|^\ell = k$.

[22] More generally, for any code C that is locally testable and decodable (such as the LDE code), one can transform an HIP for the language $\mathcal{L}$ into an IPP for the language $C(\mathcal{L}) = \{C(x) : x \in \mathcal{L}\}$. Moreover, if the query location produced by the HIP verifier is uniformly distributed (which is typically the case), then local testability by itself suffices.

Recall that for a given string $x \in \{0,1\}^k$, we define $\mathsf{LDE}_{\mathbb{F},H,m}$ as the unique individual degree $|H|-1$ polynomial $P : \mathbb{F}^m \to \mathbb{F}$ such that $P(z) = x_z$, for every $z \in H^m$ (where we identify the sets H^m and $[k]$ in some, computationally efficient, canonical way). Similarly, we define $\mathsf{LDE}^{\mathbb{K}}_{G,\ell}$ as the unique individual degree $|G|-1$ polynomial $P : \mathbb{K}^\ell \to \mathbb{K}$ such that $P(z) = x_z$, for every $z \in G^\ell$ (where now we identify G^ℓ and $[k]$).

▶ **Lemma 12.** *Let Let Let $\mathbb{F}$ and $\mathbb{K}$ be finite field ensembles as defined above. Let $\mathcal{L} = \cup_{k\in\mathbb{N}}\mathcal{L}_k$ be a language such that $\mathcal{L}_k \subseteq \{0,1\}^k$ for every $k \in \mathbb{N}$. Suppose that $\mathcal{L}$ has a ρ-round* HIP*, with respect to the code* $\mathsf{LDE}_{\mathbb{K},G,r}$*, with soundness error $\delta = \delta(k) \in [0,1]$ and communication complexity $c = c(k)$. Then, for every parameter $r = r(k) \geq 1$, the language $\mathcal{L}$ also has a $(\rho + r + 1)$-round* HIP*, with respect to the code* $\mathsf{LDE}_{\mathbb{F},H,m}$*, with soundness error $\left(\delta + O\left(\frac{|H|\cdot m}{|\mathbb{F}|}\right)\right)$ and communication $\left(c + \mathsf{poly}(k^{1/r}, |H|, r, \log|\mathbb{F}|)\right)$.*

Furthermore, the computational overhead for the verifier is $\mathsf{poly}(k^{1/r}, |H|, r, \log|\mathbb{F}|)$ and the computational overhead for the prover is $\mathsf{poly}(k)$.

We remark that for the furthermore part, we make use of the [45] constant-round variant of the GKR protocol.

Proof of Lemma 12. Before presenting the desired HIP, we start with some algebraic notation and basic facts. Throughout this proof we use $\langle\cdot,\cdot\rangle_{\mathbb{K}}$ and $\langle\cdot,\cdot\rangle_{\mathbb{F}}$ to denote inner products over the fields $\mathbb{K}$ and $\mathbb{F}$, respectively.

Recall that elements in $\mathbb{K}$ are represented as vectors in $\mathbb{F}^s$. Let $b_1, \ldots, b_s : \mathbb{K}^* \to \mathbb{F}^*$ be functions defined as follows. For every $\alpha \in \mathbb{K}^*$ it holds that $\alpha = (b_1(\alpha), \ldots, b_s(\alpha))$. That is, the functions $b_1, \ldots, b_s$ decompose a vector $w \in \mathbb{K}^t$ into its s components over $\mathbb{F}^t$.

▶ **Proposition 13.** *For every $w \in \mathbb{K}^k$ and $x \in \{0,1\}^k$ it holds that*

$$\langle w, x\rangle_{\mathbb{K}} = (\langle b_1(w), x\rangle_{\mathbb{F}}, \ldots, \langle b_s(w), x\rangle_{\mathbb{F}}).$$

Proof. We denote by $*$ multiplication in $\mathbb{K}$ and by $\cdot$ multiplication in $\mathbb{F}$. For $k = 1$ the proposition simply states that, for $w \in \mathbb{K}$ and $x \in \{0,1\}$ it holds that $w * x = (b_1(w) \cdot x, \ldots, b_s(w) \cdot x)$. The latter can be easily verified to hold for $x \in \{0,1\}$ by observing that, in both $\mathbb{K}$ and $\mathbb{F}$, multiplication by $x = 0$ always returns 0 and multiplication by $x = 1$ is identity. The proposition follows by induction on k. ◀

We proceed to describe the HIP $(\mathcal{P}', \mathcal{V}')$. Let $(\mathcal{P}, \mathcal{V})$ be an HIP for $\mathcal{L}$, with respect to the code $\mathsf{LDE}_{\mathbb{K},G,r}$, with soundness error δ. To prove the lemma, we need to construct an HIP $(\mathcal{P}', \mathcal{V}')$ for $\mathcal{L}$, with respect to the code $\mathsf{LDE}_{\mathbb{F},H,m}$.

First, $\mathcal{P}'$ and $\mathcal{V}'$ emulate the HIP $(\mathcal{P}, V)$. If $\mathcal{V}$ rejects, then $\mathcal{V}'$ immediately rejects. Otherwise, $\mathcal{V}$ outputs a pair $(z, \nu) \in \mathbb{K}^\ell \times \mathbb{K}$ with the associated claim that $\left(\mathsf{LDE}^{\mathbb{K}}_{G,\ell}(x)\right)|_z = \nu$. Since $\mathsf{LDE}^{\mathbb{K}}_{G,\ell}$ is a $\mathbb{K}$-linear code, there exists a vector $w \in \mathbb{K}^k$ (that depends only on the code $\mathsf{LDE}^{\mathbb{K}}_{G,\ell}$ and the point z) such that $\left(\mathsf{LDE}^{\mathbb{K}}_{G,\ell}(x)\right)|_z = \langle w, x\rangle$, for every $x \in \{0,1\}^k$. Thus, $\mathcal{V}'$ only needs to verify that $\langle w, x\rangle_{\mathbb{K}} = \nu$.

For every $i \in [s]$, let $w_i \stackrel{\text{def}}{=} b_i(w) \in \mathbb{F}^k$ and let $\nu_i \stackrel{\text{def}}{=} b_i(\nu) \in \mathbb{F}$. By Proposition 13 the $\mathbb{K}$-linear equation $\langle w, x\rangle = \nu$ is equivalent to the following s $\mathbb{F}$-linear equations:

$$\forall i \in [s], \quad \langle w_i, x\rangle_{\mathbb{F}} = \nu_i. \tag{5}$$

The verifier $\mathcal{V}'$ chooses at random an $\mathbb{F}$-linear combination of these s linear equations. Namely, it selects at random $\gamma_1, \ldots, \gamma_s \in \mathbb{F}$ and sends these coefficients to the prover. Let $w' \stackrel{\text{def}}{=} \sum_{i\in[s]} \gamma_i \cdot w_i$ and $\nu' \stackrel{\text{def}}{=} \sum_{i\in[s]} \gamma_i \cdot \nu_i$ (where the summations are over $\mathbb{F}$). Note that if

Equation (5) holds then (with probability 1) $\langle w', x\rangle_{\mathbb{F}} = \nu'$, whereas if Equation (5) does not hold then $\langle w', x\rangle_{\mathbb{F}} \neq \nu'$ with probability $1 - \frac{1}{|\mathbb{F}|}$ over the choice of $\gamma_1, \dots, \gamma_s \in \mathbb{F}$. We next observe that the latter is an $\mathbb{F}$-linear claim about the input x and such claims can be directly solved using the sumcheck protocol.

Let $\tilde{x} : \mathbb{F}^m \to \mathbb{F}$ (resp., $\tilde{w'}$) be the low degree extension of the input x (resp., the vector $w' \in \mathbb{F}^k$) with respect to the field $\mathbb{F}$, set H and dimension m. That is, $\tilde{x}$ and $\tilde{w'}$ are the unique individual degree $|H| - 1$ polynomial that agree with x and w', respectively, on H^m. Let $P : \mathbb{F}^m \to \mathbb{F}$ be defined as the individual degree $2(|H|-1)$ polynomial $P(z) = \tilde{w'}(z) \cdot \tilde{x}(z)$. Note that $\sum_{z \in H^m} P(z) = \langle w', x\rangle_{\mathbb{F}}$. Thus, checking that $\langle w', x\rangle_{\mathbb{F}} = \nu'$ is equivalent to $\sum_{z \in H^m} P(z) = \nu'$ which we can solve by having the prover and verifier run the sumcheck protocol with respect to the polynomial P.[23]

In case the sumcheck verifier rejects then $\mathcal{V}'$ immediately rejects. Otherwise, the result is a pair $(z'', \nu'') \in \mathbb{F}^m \times \mathbb{F}$. The prover sends to the verifier the value $\mu = \tilde{x}(z'')$. The verifier $\mathcal{V}'$ checks that $\mu \cdot \tilde{w'}(z'') = \nu''$ and if so it outputs (z'', μ), otherwise it rejects. This completes the description of the protocol.

Actually, one point about this protocol remains unclear - how can the verifier efficiently compute $\tilde{w'}(z'')$. If we were to ignore the *computational* resources of the verifier, then we could do this by brute force (e.g., in time roughly $|H|^m$), since $\tilde{w'}$ is independent of the input x. Nevertheless, we do aim for efficient verification and so we need to be able to compute $\tilde{w'}(z'')$ efficiently. We will do so by using additional interaction with the prover, based on the [45] variant of the GKR protocol. We give a sketch in the following paragraph.

Computing $\tilde{w'}(z'')$. We start by taking a closer look at the vector w defined above. By the definition of the low degree extension (see Section 2.3), the vector $w \in \mathbb{K}^{G^\ell}$ is defined as $w_h = \beta(h, z)$, for every $h \in G^\ell$, where β is as defined in Section 2.3. Thus, we have that:

$$\tilde{w'}(z'') = \sum_{i \in [s]} \gamma_i \cdot b_i \left(\sum_{h \in H^m} \beta(h, z'') \cdot \beta(h, z) \right). \tag{6}$$

We observe that Equation (6) can be represented as a (highly uniform) depth $O(\log(s) + m \cdot \log(H) + \log(|G|) + \log(\ell)) = O(\log(k))$ Boolean circuit (on input z, z'') of size $s \cdot H^m \cdot \mathsf{poly}(|G|, \ell, \log(|\mathbb{K}|)) = \mathsf{poly}(k)$. Applying the [45] variant of the GKR protocol, we obtain an r-round interactive proof for verifying Equation (6) in which the verifier runs in time $k^{O(1/r)} \cdot \mathsf{polylog}(|\mathbb{F}|)$ and with similar communication complexity.

Completeness. Fix $x \in \mathcal{L}$. By the completeness of $(\mathcal{P}, \mathcal{V})$, the verifier outputs $(z, \nu) \in \mathbb{K}^\ell \times \mathbb{K}$ such that $\left(\mathsf{LDE}^{\mathbb{K}}_{G,\ell}(x)\right)\big|_z = \nu$, or equivalently, $\langle w, x\rangle_{\mathbb{K}} = \nu$. By Proposition 13 this implies that $\langle w_i, x\rangle_{\mathbb{F}} = \nu_i$, for every $i \in [s]$. Therefore, for every $\gamma_1, \dots, \gamma_s \in \mathbb{F}$ it holds that:

$$\langle w', x\rangle_{\mathbb{F}} = \sum_{i \in [s]} \gamma_i \cdot \langle w_i, x\rangle_{\mathbb{F}} = \sum_{i \in [s]} \gamma_i \cdot \nu_i = \nu'.$$

By definition of P, this means that $\sum_{z \in H^m} P(z) = \langle w', x\rangle_{\mathbb{F}} - \nu' = 0$ and the completeness of the sumcheck protocol implies that $\nu'' = P(z'') = \tilde{x}(z'') \cdot \tilde{w}(z'')$. Thus the verifier accepts when checking that $\mu \cdot \tilde{w}(z'') = \nu''$.

[23] We remark that while we defined sumcheck as a protocol for the language $\mathcal{L} = \{x \in \mathbb{F}^k : \sum_{i \in [k]} x_i = 0\}$, a trivial, standard modification of the sumcheck protocol yields a protocol for $\mathcal{L}_\nu = \{x \in \mathbb{F}^k : \sum_{i \in [k]} x_i = \nu\}$, for every $\nu \in \mathbb{F}$.

Soundness. Fix $x \notin \mathcal{L}$ and a cheating prover strategy $\mathcal{P}^*$. By the soundness of (P, V), with probability $1 - \varepsilon$, the verifier either rejects (in which case $\mathcal{V}'$ also rejects) or outputs $(z, \nu) \in \mathbb{K}^\ell \times \mathbb{K}$ such that $\langle w, x\rangle_{\mathbb{K}} \neq \nu$. Assuming that the latter holds, by Proposition 13 there exists some $i^* \in [s]$ such that $\langle w_{i^*}, x\rangle_{\mathbb{F}} \neq \nu_{i^*}$. Therefore,

$$\begin{aligned}\Pr\left[\langle w', x\rangle_{\mathbb{F}} = \nu'\right] &= \Pr\left[\sum_{i\in[s]} \gamma_i \cdot \langle w_i, x\rangle_{\mathbb{F}} = \sum_{i\in[s]} \gamma_i \cdot \nu_i\right] \\ &= \Pr\left[\gamma_{i^*} \cdot (\langle w_{i^*}, x\rangle_{\mathbb{F}} - \nu_{i^*}) = \sum_{i\neq i^*} \gamma_i \cdot (\nu_i - \langle w_i, x\rangle_{\mathbb{F}})\right] \\ &= 1/|\mathbb{F}|.\end{aligned}$$

Thus, with probability $1 - \frac{1}{|\mathbb{F}|}$ it holds that $\langle w', x\rangle_{\mathbb{F}} \neq \nu'$, and in particular $\sum_{z\in H^m} P(z) \neq 0$ (where P is the polynomial as defined above).

Hence, by the soundness of the sumcheck protocol, with probability $\frac{|H|\cdot m}{|\mathbb{F}|}$ either the sumcheck verifier rejects (in which case we also reject) or it outputs a pair $(z'', \nu'') \in \mathbb{F}^m \times \mathbb{F}$ such that $P(z'') \neq \nu''$, or in other words $\tilde{x}(z'') \cdot \tilde{w}(z'') \neq \nu''$. Now, the prover sends over a value μ. If $\tilde{x}(z'') = \mu$ then, conditioned on the above event, the verifier rejects when checking that $\mu \cdot \tilde{w}(z'') = \nu'$. If $\tilde{x}(z'') \neq \mu$, then the verifier outputs a pair (z'', μ) such that $\left(\mathsf{LDE}_{\mathbb{F},H,m}(x)\right)_{z''} \neq \mu$ as desired. By a union bound, the overall soundness error is $\varepsilon + \frac{1}{|\mathbb{F}|} + \frac{|H|\cdot m}{|\mathbb{F}|}$.

Complexity. On top of the ρ rounds that $(\mathcal{P}, \mathcal{V})$ takes, the verifier also sends the message $(\gamma_1, \ldots, \gamma_s)$, but this message can be appended to the last message from $\mathcal{V}$ to $\mathcal{P}$. In addition, the two parties run an r-round sumcheck protocol and an r round variant of the GKR protocol. There is one additional message from the prover with the value μ, so the overall number of rounds is $\rho + O(r)$.

The communication in the first part of the protocol (i.e., the emulation of (P, V)) is c. In addition, the verifier sends the linear combination $(\gamma_1, \ldots, \gamma_s)$ which takes $s \cdot \log|\mathbb{F}|$ bits. Lastly, both the sumcheck and the GKR protocol add communication $\mathsf{poly}(k^{1/r}, |H|, r, \log|\mathbb{F}|)$ and the additional prover message is just $\log_2|\mathbb{F}|$ bits.

As for the verifier's complexity, beyond running the original $(\mathcal{P}, \mathcal{V})$ protocol, it runs the sumcheck and GKR protocols which takes time $\mathsf{poly}(k^{1/r}, |H|, r, \log|\mathbb{F}|)$. The prover's additional time in running these two protocols is $\mathsf{poly}(|H|^m) = \mathsf{poly}(k)$. ◀

3.4 Booleanity Testing

In this subsection we show that HIPs can efficiently check that their input is the low-degree extension of a *Boolean* assignment. To do so, we follow an idea of Kalai and Raz [46], which was introduced in the context of constructing interactive PCPs.

We show a simple reduction from checking whether a polynomial $P : \mathbb{F}^m \to \mathbb{F}$ is Boolean valued in a subcube H^m (i.e., $P|_{H^m} H^m \to \{0, 1\}$) to checking whether a related (slightly higher degree) polynomial Q vanishes on H^m. Specifically, consider the polynomial $Q(x) = P(x) \cdot (1 - P(x))$, and observe that P is Boolean-valued in H^m if only if Q is identically zero in H^m. Checking whether a given polynomial is identically 0 (i.e., vanishes) on a subcube of its domain can be solved via a fairly well-known reduction to the sumcheck protocol. We also note that the reduction from P to Q is local (i.e., each query to Q can be computed by a single query to P) and therefore can be used in our setting.

We start by showing an HIP for inputs that vanish on a subcube. We first note that checking whether an individual degree $|H|-1$ polynomial vanishes on the subcube H^m is trivial, since such a polynomial vanishes on H^m if and only if it vanishes on $\mathbb{F}^m$. The actual challenge is checking whether a higher degree polynomial (e.g., with individual degree $|G|-1$ for some G such that $|G|>|H|$) vanishes on H^m.

Formally, for a given field ensemble $\mathbb{F}$, ensembles of subsets $H, G \subseteq \mathbb{F}$ and dimension m, let $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$ be the set of all functions $f : G^m \to \mathbb{F}$ that vanish on H^m (i.e., $f|_{H^m} \equiv 0$).

The following proposition, which gives an HIP for $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$, is implicit in many classical constructions of PCPs (e.g., [5]). We include a proof in Appendix D for completeness.

▶ **Proposition 14.** *Let $\mathbb{F}$ be a constructible field ensemble, let $H \subseteq G \subseteq \mathbb{F}$ be ensembles of subsets, and let $m = m(k)$. For every $r = r(k) \le \frac{\log(k)}{\log\log(k)}$, there exists an $(r+2)$-round (public-coin)* HIP *for* $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$*, with respect to the code* $\mathsf{LDE}_{\mathbb{F},G,m}$*, with soundness error $O\left(\frac{m\cdot|G|}{|\mathbb{F}|}\right)$ and communication complexity $m \cdot \log(|\mathbb{F}|) + |G|^{\lceil m/r \rceil} \cdot r \cdot \log|\mathbb{F}|$. The verifier runs in time $|G|^{\lceil m/r \rceil} \cdot r \cdot \mathsf{polylog}(|\mathbb{F}|)$ and the prover runs in time $\mathsf{poly}(|\mathbb{F}|^m)$.*

Denote by $\mathsf{Bool}_{\mathbb{F}}$ the set of all Boolean strings, viewed as a subset of $\mathbb{F}^*$. We show an HIP for Bool, which given access to a polynomial $P = \mathsf{LDE}_{\mathbb{F},H,m}(x)$ for some $x \in \mathbb{F}^k$, checks that $x \in \{0,1\}^k$.

▶ **Proposition 15.** *Let $\mathbb{F}$ be a constructible field ensemble, let $H \subseteq \mathbb{F}$, and let $m \in \mathbb{N}$. For every $r \in [m]$, there exists an $(r+2)$-round (public-coin)* HIP *for* $\mathsf{Bool}_{\mathbb{F},H,m}$*, with respect to the code* $\mathsf{LDE}_{\mathbb{F},H,m}$*, with communication complexity $O(r\cdot(2d+|H|-1)^{m/r}\cdot\log|\mathbb{F}| + m\cdot\log(|\mathbb{F}|))$ and soundness error $O\left(\frac{m\cdot|H|}{|\mathbb{F}|}\right)$.*

Proof. Given a degree d polynomial $P : \mathbb{F}^m \to \mathbb{F}$ such that $P = \mathsf{LDE}_{\mathbb{F},H,m}(x)$ for some $x \in \mathbb{F}^{|H|^m}$, define the degree $2d$ polynomial $Q : \mathbb{F}^m \to \mathbb{F}$ as $Q(x) \overset{\text{def}}{=} P(x)\cdot(1-P(x))$. Note that we can write $Q = \mathsf{LDE}_{\mathbb{F},G,m}(y)$ for $H \subseteq G \subseteq \mathbb{F}$ and $y \in \mathbb{F}^{|G|^m}$, where $|G| = O(|H|)$.

Observe that P is Boolean-valued in H^m if and only if Q is identically 0 in H^m (this follows from the fact that the univariate polynomial $z\cdot(1-z)$ has exactly two roots: 0 and 1). Thus, to verify that P is Boolean-valued in H^m, we run the HIP for $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$ in Proposition 14, with respect to the polynomial Q. Note that each query $Q(x)$ can be answered by a single query to P (specifically, by returning $P(x)\cdot(1-P(x))$). Correctness follows from the correctness of the HIP for $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$. Communication complexity and soundness error follow from Proposition 14. ◀

4 The Hierarchy Theorem

In this section we prove our main theorem: a round hierarchy for IPPs.

▶ **Theorem 16** (IPP Hierarchy Theorem). *There exists a language $\mathcal{L}$ and a gap function $g(r) = \Theta(r^2)$ such that for every constant $r \ge 1$ it holds that:*

1. **Upper Bound:** *There exists a $g(r)$-round (public-coin) ε-*IPP*, for $\mathcal{L}$ with communication complexity $n^{O(1/r)}$ and query complexity $\mathsf{poly}(\log n, \varepsilon)$. The verifier runs in time $n^{O(1/r)} + \mathsf{poly}(\log(n), \varepsilon)$ and the prover runs in time $\mathsf{poly}(n)$.*
2. **Lower Bound:** *For every r-round* IPP *for $\mathcal{L}$, with respect to proximity parameter $\varepsilon = 1/10$, that has query complexity q and communication complexity c, it holds that $\max(c, q) = n^{\Omega(1/r)}$.*

Furthermore, $\mathcal{L}$ also has a $\mathsf{polylog}(n)$*-round (public-coin)* ε-IPP *with communication* $\mathsf{polylog}(n)$ *and query complexity* $\mathsf{poly}(\log n, 1/\varepsilon)$*, and with a* $\mathsf{poly}(\log n, \varepsilon)$*-verifier and* $\mathsf{poly}(n)$*-time prover.*

The O and Ω notation in the theorem statement hide universal constants that do not depend on r. Note that any constant gap between the exponents in the upper and lower bounds can be obtained by increasing g by a suitable constant factor.

The rest of this section is devoted to the proof of Theorem 16. In Section 4.1 we present the language for which we show the IPP round hierarchy, in Section 4.3 we prove the lower bound (see Lemma 23), and in Section 4.2 we prove the upper bound (see Lemma 17). Combining Lemma 23 and Lemma 17 yields Theorem 16.

4.1 The Language: Encoded MOD3

Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be a (constructible) field ensemble of characteristic 2 (i.e., each $\mathbb{F}_k$ is an extension field of $\mathsf{GF}(2)$). Let $H = (H_k)_{k\in\mathbb{N}}$ be an ensemble of subsets $H \subseteq \mathbb{F}$ and let $m = m(k)$ be the dimension such that $|H| = \log(k)$, $m = \frac{\log(k)}{\log\log(k)}$ and $|\mathbb{F}| = \Theta(|H|^2 m)$. Denote $n \overset{\text{def}}{=} |\mathbb{F}^m|$, and note that $|H|^m = k$ and that $k^2 \leq n \leq k^3$.

We first define an (auxiliary) language $\mathcal{L}_{\mathsf{MOD3}}$, where:

$$\mathcal{L}_{\mathsf{MOD3}} \overset{\text{def}}{=} \{x \in \{0,1\}^* \;:\; \mathsf{wt}(x) = 0 \pmod 3\}\,.$$

That is, $\mathcal{L}_{\mathsf{MOD3}}$ simply consists of strings whose Hamming weight is divisible by 3. The actual language for which we prove the IPP lower bound is $\mathsf{Enc\text{-}MOD3} = \mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L}_{\mathsf{MOD3}})$. That is,

$$\mathsf{Enc\text{-}MOD3} = \left\{\mathsf{LDE}_{\mathbb{F},H,m}(x) \;:\; x \in \mathcal{L}_{\mathsf{MOD3}}\right\}.$$

Or in words, Enc-MOD3 consists of all m-variate polynomials over $\mathbb{F}$, of individual degree $|H| - 1$, that take Boolean values in H^m such that the integer sum over all elements in H^m is divisible by 3.

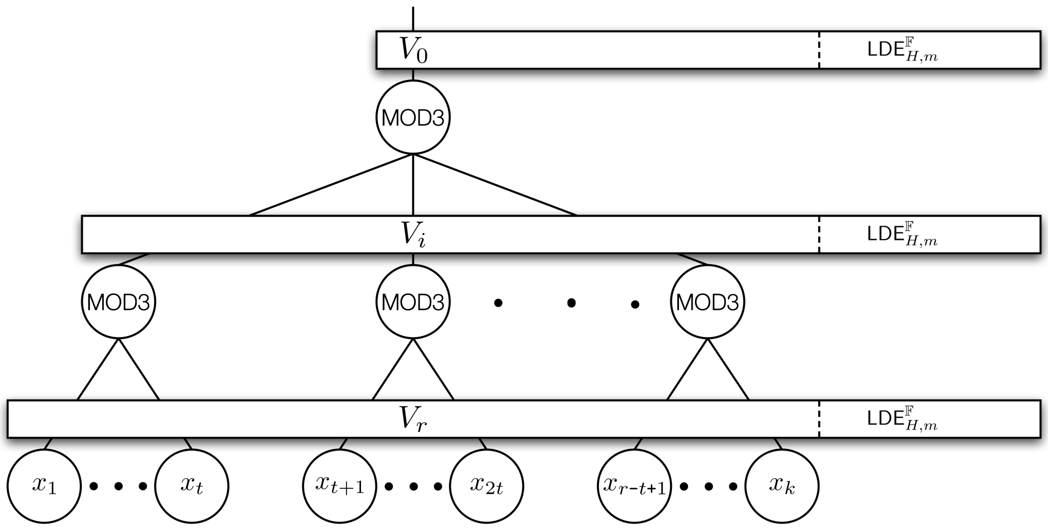
4.2 The Upper Bound

In this section, we construct an IPP for Enc-MOD3. This IPP suffices both for the results in the constant-round regime and poly-logarithmic round regime of Theorem 16.

▶ **Lemma 17.** *For every* $r = r(n) \leq \frac{\log(n)}{\log\log(n)}$*, there exists an* $O(r^2)$*-round public-coin* ε-IPP *for* $\mathsf{Enc\text{-}MOD3}$ *with perfect completeness and soundness error* $1/2$*. The communication complexity is* $n^{O(1/r)}$ *and the query complexity is* $\mathsf{poly}(\log(n), 1/\varepsilon)$*. Furthermore, the verifier runs in time* $\left(n^{O(1/r)} + \mathsf{poly}(\log(n), 1/\varepsilon)\right)$ *and the prover runs in time* $\mathsf{poly}(n)$*.*

The main step in the proof of Lemma 17 is the construction of an HIP for the related language $\mathcal{L}_{\mathsf{MOD3}}$ (defined above), with respect to the LDE code (with the parameters that were specified in Section 4.1). Given this HIP, Lemma 17 follows by using a generic transformation from HIPs (with respect to the LDE encoding) into IPPs, which we establish in Proposition 11.

Before constructing this HIP, as an intermediate goal, we first construct an HIP for $\mathcal{L}_{\mathsf{MOD3}}$, with respect to the low-degree extension with different parameters than those that were set in Section 4.1. Specifically, we shall use a larger field $\mathbb{K}$, whose size is polynomially related to k (rather than poly-logarithmic). In particular, there will be a dependence between the size of $\mathbb{K}$ and the number of rounds in the HIP. Later we will use a generic transformation to



V_0
$\mathsf{LDE}^{\mathbb{F}}_{H,m}$
MOD3
V_i
$\mathsf{LDE}^{\mathbb{F}}_{H,m}$
MOD3
MOD3
MOD3
V_r
$\mathsf{LDE}^{\mathbb{F}}_{H,m}$
x_1
x_t
x_{t+1}
x_{2t}
x_{r-t+1}
x_k

Figure 1 The recursive depth r formula of fan-in $k^{1/r}$ that computes the sum mod 3 of its input $x \in \{0,1\}^k$, and the low-degree extension of each one of the formula's layers when evaluated on x.

convert this HIP into one in which the low degree extension can be over a much smaller field (e.g., of poly-logarithmic size), which in particular does not depend on the number of rounds.

The following lemma, which is the main lemma proved in this section, gives an HIP for $\mathcal{L}_{\mathsf{MOD3}}$ over the relatively large field $\mathbb{K}$.

▶ **Lemma 18.** *Let $r = r(k) \geq 1$, let $\mathbb{K} = (\mathbb{K}_k)_{k\in\mathbb{N}}$ be a constructible field ensemble of size $|\mathbb{K}| = \Omega(r^2 \cdot k^{2/r})$, let $G = (G_k)_{k\in\mathbb{N}} \subseteq \mathbb{K}$ be an ensemble of subsets of $\mathbb{K}$ of size $|G| = k^{1/r}$.*

Then, there exists an r^2-round public-coin HIP for $\mathcal{L}_{\mathsf{MOD3}}$, with perfect completeness and soundness error $O\left(\frac{r^2 \cdot k^{2/r}}{|\mathbb{K}|}\right)$. The communication complexity is $O(r^2 \cdot k^{2/r} \cdot \log|\mathbb{K}|)$. The verifier runs in time $k^{O(1/r)} \cdot \mathsf{poly}(r, \log(k))$ and the prover runs in time $\mathsf{poly}(|\mathbb{K}|^r)$.

(See Section 1.2 for a high-level overview of the proof.)

Proof. Let $r = r(k) \geq 1$. Recall that $\mathbb{K} = (\mathbb{K}_k)_{k\in\mathbb{N}}$ is a constructible field ensemble field of size $|\mathbb{K}| = \Omega(r^2 \cdot k^{1/r})$ and that $G = (G_k)_{k\in\mathbb{N}}$ is an ensemble of subsets of size $|G| = k^{1/r}$. Since we only deal with a single input length k (which we think of as varying), in the following we omit the subscripts and use $\mathbb{K}$ (resp., G) when we actually mean $\mathbb{K}_k$ (resp., G_k).

Denote by $t \stackrel{\text{def}}{=} |G| = k^{1/r}$ and fix a canonical ordering $\alpha_1, \ldots, \alpha_t$ of the set of elements in G (i.e., $G = \{\alpha_1, \ldots, \alpha_t\}$). Let $\mathsf{MOD3}_t : \{0,1,2\}^t \to \{0,1,2\}$ be defined as $\mathsf{MOD3}_t(\sigma_1, \ldots, \sigma_t) \stackrel{\text{def}}{=} \sum_{j\in[t]} \sigma_j \pmod 3$.

Fix an input $x \in \{0,1\}^k$. As described in Section 1.2, we define polynomials $V_0, \ldots, V_r$ that contain sums, modulo 3, of certain intervals in x. Taking the [37] view, one can consider a depth r formula, with fan-in $t = k^{1/r}$, composed of $\mathsf{MOD3}_t$ gates, that computes the sum mod 3 of its input (see Figure 1). Viewed this way, each polynomial V_i corresponds to the low degree extension of the i^{th} layer of this formula (counting from output to input).

Since $|G^r| = k$, we can associate elements in G^r with the integers in the set $\{1, \ldots, k\}$ in the natural way. Thus, we can view the input $x \in \{0,1\}^k$ as a function, which we denote by $V_r : G^r \to \{0,1\}$, that is defined as $V_r(p) = x_p$, for every $p \in G^r$. We define functions $V_0, \ldots, V_{r-1}$ via backward recursion as follows. For every $i \in [r]$, let $V_{i-1} : G^{i-1} \to \{0,1,2\}$

be defined as:

$$\forall p \in G^{i-1}, \quad V_{i-1}(p) = \mathsf{MOD3}_t\big(V_i((p,\alpha_1)),\ldots,V_i((p,\alpha_t))\big), \tag{7}$$

where (p,α) denotes the element in G^i which is obtained by concatenating $p \in G^{i-1}$ with $\alpha \in G$. For the case $i = 0$, we define $G^0 = \{\perp\}$, where $\perp$ is defined as the empty string (in particular $(\perp,p) = (p,\perp) = p$), and note that, for $i = 1$, Equation (7) reduces to $V_0(\perp) = \mathsf{MOD3}_t\left(V_1(\alpha_1)\right),\ldots,V_1(\alpha_t)\big)$.

As noted above, intuitively, each V_i should be thought of as specifying a sum of certain intervals in the input, according to a partition (which depends on i). For example, V_r contains the value of each of the individual coordinate of x (i.e., the most fine grained partition) whereas V_0 contains the overall sum (i.e., the coarsest partition). More generally, we have the following immediate fact:

▶ **Fact 19.** *For every $i \in \{0,\ldots,r\}$ and $p \in G^i$ it holds that $V_i(p) = \sum_{q\in G^{r-i}} x_{(p,q)} \pmod 3$ (where (p,q) denotes the concatenation of the two vectors p and q).*

In particular, by setting $i = 0$, we have that $V_0(\perp) = \sum_{q\in G^r} x_q \pmod 3 = \sum_{i\in[k]} x_i \pmod 3$.

For the rest of the proof we use $\tilde{f}$ to denote the low degree extension of a function f (see Section 2.3 for details on the low degree extension encoding) and associate the integers 0, 1 and 2 with three distinct elements in $\mathbb{K}$ in some canonical way (so that we can view $\{0,1,2\} \subseteq \mathbb{K}$). Let $\widetilde{\mathsf{MOD3}}_t : \mathbb{K}^t \to \mathbb{K}$ be the unique individual degree 2 extension of the function $\mathsf{MOD3} : \{0,1,2\}^t \to \{0,1,2\}$ with respect to the field $\mathbb{K}$, the subset $\{0,1,2\} \subseteq \mathbb{K}$, and dimension t. For every $i \in [r]$, let $\tilde{V}_i : \mathbb{K}^i \to \mathbb{K}$ be the unique individual degree $|G| - 1$ extension of V_i with respect to the field $\mathbb{K}$, the set G and dimension i. Let $\tilde{V}_0 \equiv V_0$ (recall that $V_0 : \{\perp\} \to \{0,1,2\}$ is just a singleton value $V_0(\perp) \in \mathbb{K}$). Observe that the polynomial $\tilde{V}_r$ is the low degree extension of the input x with respect to the field $\mathbb{K}$, the set G and dimension r.

A crucial fact that we will use is that, for every $i \in [r]$, each point in $\tilde{V}_{i-1}$ can be expressed as a certain type of composition of the low degree polynomial $\tilde{V}_i$ with the low degree polynomial $\widetilde{\mathsf{MOD3}}$. More specifically, using the properties of the low degree extension (see Section 2.3), it holds that for every $i \in [r]$ and $z \in \mathbb{K}^{i-1}$:

$$\begin{aligned}\tilde{V}_{i-1}(z) &= \sum_{p\in G^{i-1}} \beta(z,p)\cdot V_{i-1}(p)\\ &= \sum_{p\in G^{i-1}} \beta(z,p)\cdot \mathsf{MOD3}_t\big(V_i((p,\alpha_1)),\ldots,V_i((p,\alpha_t))\big)\\ &= \sum_{p\in G^{i-1}} \beta(z,p)\cdot \widetilde{\mathsf{MOD3}}_t\big(\tilde{V}_i((p,\alpha_1)),\ldots,\tilde{V}_i((p,\alpha_t))\big). \end{aligned}\tag{8}$$

where the polynomial β is as defined in Section 2.3, and the last equality uses the fact that $\tilde{V}_i|_{G^i} \equiv V_i|_{G^i}$ and $\widetilde{\mathsf{MOD3}}_t|_{\{0,1,2\}^t} \equiv \mathsf{MOD3}_t|_{\{0,1,2\}^t}$.

Using the above definition, we proceed to describe our HIP for $\mathcal{L}_{\mathsf{MOD3}}$. The protocol is performed in r phases, each of which takes at most r rounds of interaction (for a total of at most r^2 rounds). We begin the protocol with a claim about the value of a single point (as a matter of fact, the only point) in $\tilde{V}_0$ (recall that, by Fact 19, the value of $\tilde{V}_0(\perp)$ corresponds to the desired output - the sum modulo 3 of the input bits). In the i^{th} phase, we reduce the task of verifying the value of a single (arbitrary) point in $\tilde{V}_{i-1}$ to verifying the value of a single point in $\tilde{V}_i$. Thus, after r phases, we have reduced the problem of verifying $\tilde{V}_0(\perp) = \sum_{j\in[k]} x_j \pmod 3$ to verifying a single point in $\tilde{V}_r$, which is the low degree extension of the input x.

Define $z_0 = \perp$ and $\nu_0 = 0$. The original claim is that $\tilde{V}_r(z_0) = \nu_0$. We shall maintain the invariant that for every phase $i \in \{0,\dots,r\}$, at the end of the i^{th} phase, the prover and verifier both know a vector $z_i \in \mathbb{K}^i$ and a scalar $\nu_i \in \mathbb{K}$ such that the current claim is that $\tilde{V}_i(z_i) = \nu_i$. Thus, the goal of the i^{th} phase is to (interactively) reduce the claim $\tilde{V}_{i-1}(z_{i-1}) = \nu_{i-1}$ to a claim of the form $\tilde{V}_i(z_i) = \nu_i$ (for some z_i and ν_i that are generated during the i^{th} phase):

Phase i.

1. **Reduce to Claim about t Points in $\tilde{V}_i$:** The phase begins with a claim that $\nu_{i-1} = \tilde{V}_{i-1}(z_{i-1})$. By Equation (8) this is equivalent to:
$$\nu_{i-1} = \sum_{p\in G^{i-1}} \beta(z_{i-1},p)\cdot \widetilde{\mathsf{MOD3}}_t\big(\tilde{V}_i((p,\alpha_1)),\dots,\tilde{V}_i((p,\alpha_t))\big) \tag{9}$$
We now observe that the right-hand side of Equation (9) corresponds to a sum, over an $(i-1)$-dimensional subcube, of the values of a low degree polynomial. Specifically, denote
$$f_{i-1}(w) = \beta(z_{i-1},w)\cdot \widetilde{\mathsf{MOD3}}_t\big(\tilde{V}_i((w,\alpha_1)),\dots,\tilde{V}_i((w,\alpha_t))\big),$$
and observe that f_{i-1} has *total* degree $(t-1)\cdot(i-1)+2t\cdot(t-1)\cdot i \le 3t^2 r$ polynomial. Equation (9) can be rewritten as $\nu_{i-1} = \sum_{p\in G^{i-1}} f_{i-1}(p)$. The prover and verifier run an i-round sumcheck protocol with respect to this equation.
In case the sumcheck verifier rejects, our verifier immediately rejects. Otherwise, the output of the sumcheck protocol is a (random) point $w_{i-1} \in \mathbb{K}^{i-1}$ and value $\gamma_{i-1} \in \mathbb{K}$ with an associated alleged claim that $\gamma_{i-1} = f_{i-1}(w_{i-1})$.
2. **Query Reduction:** At this point the verifier has a claim regarding the values of t points of $\tilde{V}_i$ (specifically, the claim $\gamma_{i-1} = f_{i-1}(w_{i-1})$ refers to the points $(w_i,\alpha_1),\dots,(w_i,\alpha_t)$). The goal of this step is to reduce this more elaborate claim to a claim about a *single* point in $\tilde{V}_i$:
   a. The prover sends to the verifier the univariate degree $t-1$ polynomial $P_i : \mathbb{K} \to \mathbb{K}$ defined as $P_i(\eta) = \tilde{V}_i(w_i,\eta)$ (given by its t coefficients).
   b. The verifier receives a degree $t-1$ polynomial Q_i (which is allegedly equal to P_i). The verifier checks that $\gamma_i = \beta(z_{i-1},w_{i-1})\cdot \widetilde{\mathsf{MOD3}}_t\big(Q_i(\alpha_1),\dots,Q_i(\alpha_t)\big)$. If the check fails then the verifier immediately rejects and halts. Otherwise, the verifier chooses a random field element $\eta_i \in \mathbb{K}$ and sends η_i to the prover.
   c. The claim for the next round is that $\tilde{V}_i(z_i) = \nu_i$, where $\nu_i = P_i(\eta_i)$ and $z_i = (w_i,\eta_i)$.

After all of the r phases are complete, the verifier outputs (z_r,ν_r) and the associated claim is that $\tilde{V}_r(z_r) = \nu_r$. Since $\tilde{V}_r$ is simply the low degree extension of the input x, the latter is a claim about a single point in the low degree extension of the input as required by the definition of an HIP verifier.

Complexity. Since the communication complexity of each sumcheck is $O(r\cdot k^{1/r}\cdot\log|\mathbb{K}|)$, the total communication complexity is $O(r^2\cdot k^{1/r}\cdot\log|\mathbb{K}|)$. As for the round complexity, the i^{th} phase uses a sumcheck of $i \le r$ rounds of interaction. Moreover, each sumcheck concludes with a message from the prover to the verifier so we can "piggyback"and attach the polynomial Q_i to that last message from the prover and send back the value η_i as our response (which is still part of the last round of the sumcheck protocol) so each phase just takes $\le r$ rounds and overall we have $\le r^2$ rounds.

As for computational complexity, in the first step of each phase, the parties invoke a sumcheck protocol in which, by Lemma 10, the verifier runs in time $k^{O(1/r)} \cdot r \cdot \mathsf{polylog}|\mathbb{K}|$, and the prover runs in time $\mathsf{poly}(|\mathbb{K}|^r)$. In the second step of each phase, the prover computes and sends P_i, which clearly can be done in time $\mathsf{poly}(|\mathbb{K}|^r)$, and the verifier computes γ_i, which boils down to evaluating the functions β and $\widetilde{\mathsf{MOD3}}_t$ at a single point, which can be done in time $\mathsf{poly}(t, \log k) = k^{O(1/r)} \cdot \mathsf{poly}(\log |\mathbb{K}|)$ (see Proposition 6 and Appendix E for the time complexity of computing β and $\widetilde{\mathsf{MOD3}}_t$, respectively). The obtain the total running times (for the entire r phases), we multiply the time per phase by r.

Completeness. Perfect completeness follows readily from the construction (and the prefect completeness of the sumcheck protocol).

Soundness. To conclude the proof of Lemma 18 we only need to show that soundness holds. Our analysis follows the soundness analysis in [37, Theorem 3.1].

Fix an input $x \in \{0,1\}^k$ such that $\sum_{i\in[k]} x_i \not\equiv 0 \pmod 3$ (i.e. $x \notin \mathcal{L}_{\mathsf{MOD3}}$) and a cheating strategy $\mathcal{P}^*$. Denote by A the event that the verifier does not reject in the interaction with the prover $\mathcal{P}^*$. For every $i \in \{0, 1, \ldots, r\}$, denote by T_i the event that $\tilde{V}_i(z_i) = \nu_i$. Note that since $\sum_{i\in[k]} x_i \neq 0 \pmod 3$ it holds that the event $\neg T_0$ occurs with probability 1. For every $i \in [r]$, let E_i denote the event that the polynomial Q_i that the prover sent is indeed identical to $P_i(\eta) = \tilde{V}_i(w_i, \eta)$.

Our analysis will be based on the following two claims.

▶ **Claim 20.**

$$\Pr\left[A \wedge E_i \mid \neg T_{i-1}\right] \leq \frac{3t^2 r}{|\mathbb{K}|}.$$

Proof. Assume that the event T_{i-1} occurs. Then, by the soundness of the sumcheck protocol, with probability $\frac{3t^2 r}{|\mathbb{K}|}$ (over the verifier's coins in the sumcheck protocol) it holds that $f_{i-1}(w_{i-1}) \neq \gamma_{i-1}$, or in other words $\beta(z_{i-1}, w_{i-1}) \cdot \widetilde{\mathsf{MOD3}}_t\big(\tilde{V}_i((w_{i-1}, \alpha_1)), \ldots,$ $\tilde{V}_i((w_{i-1}, \alpha_t))\big) \neq \gamma_{i-1}$. If the latter happens and then the prover sends the correct polynomial P_i (i.e., the event E_i occurs) then the verifier immediately rejects in Item 2b. Thus, with probability $1 - \frac{3t^2 r}{|\mathbb{K}}$, either the event $\neg A$ or $\neg E_i$ must occur. ◀

On the other hand:

▶ **Claim 21.**

$$\Pr\left[T_i \mid \neg E_i\right] \leq \frac{t}{|\mathbb{K}|}.$$

Proof. The event $\neg E_i$ implies that the polynomial Q_i sent by the prover differs from the correct polynomial P_i. Since both Q_i and P_i are degree $t-1$ polynomials, they can agree on at most $t-1$ points, and so, with probability $1 - \frac{t-1}{|\mathbb{K}|}$ over the choice of η_i it holds that $\nu_i = Q(\eta_i) \neq P(\eta_i) = \tilde{V}_i((w_i, \eta_i)) = \tilde{V}_i(z_i)$. ◀

Finally, observe that the probability that the verifier errs is simply $\Pr[A \wedge \neg T_r]$, which we can bound (using Claim 20, Claim 21 and elementary probability theory) as follows:

$$\begin{aligned}
\Pr[A \wedge T_r] &= \Pr[A \wedge \neg T_0 \wedge T_r] \\
&\leq \Pr\left[\exists i \in [r] \text{ such that } A \wedge \neg T_{i-1} \wedge T_i\right] \\
&\leq \sum_{i=1}^{r} \Pr\left[A \wedge \neg T_{i-1} \wedge T_i\right] \\
&= \sum_{i=1}^{r} \left(\Pr[A \wedge \neg T_{i-1} \wedge T_i \wedge E_i] + \Pr[A \wedge \neg T_{i-1} \wedge T_i \wedge \neg E_i]\right) \\
&\leq \sum_{i=1}^{r} \left(\Pr\left[A \wedge E_i \mid \neg T_{i-1}\right] + \Pr\left[T_i \mid \neg E_i\right]\right) \\
&\leq \sum_{i=1}^{r} \left(\frac{3t^2 r}{|\mathbb{K}|} + \frac{t}{|\mathbb{K}|}\right) \\
&\leq \frac{4t^2 r^2}{|\mathbb{K}|}.
\end{aligned}$$

This concludes the proof of Lemma 18. ◀

Lemma 18 provides an r^2-round HIP for $\mathcal{L}_{\mathsf{MOD3}}$, with respect to the code $\mathsf{LDE}_{\mathbb{K},G,r}$, where $\mathbb{K}$ is a field ensemble of size $\Theta\left(r^2 \cdot k^{2/r}\right)$. We now use a general result, which is stated and proved in Section 3, which transforms any such HIP, in which the field $\mathbb{K}$ has small characteristic, into an HIP over the code $\mathsf{LDE}_{\mathbb{F},H,m}$ where the size of the field $\mathbb{F}$ is now only *poly-logarithmic* in k. Specifically, by applying Lemma 12 to the protocol of Lemma 18, and using a field $\mathbb{K}$ which is an extension field of some field $\mathbb{F}$ of size $\mathsf{polylog}(k)$, we obtain the following corollary:

▶ **Corollary 22.** *Let $\mathbb{F} = (\mathbb{F}_k)_{k\in\mathbb{N}}$ be a constructible field ensemble, $H = (H_k)_{k\in\mathbb{N}} \subseteq \mathbb{F}$ be an ensembles of subsets of $\mathbb{F}$ and let $m = m(k)$ be a dimension such that $|H| = \log(k)$, $m = \frac{\log(k)}{\log\log(k)}$ and $|\mathbb{F}| = \Theta(|H| \cdot m)$.*

Then, for every parameter $r = r(k) \leq \frac{\log(k)}{\log\log(k)}$, the language $\mathcal{L}_{\mathsf{MOD3}}$ has an has an $O(r^2)$-round (public-coin) HIP with respect to the code $\mathsf{LDE}_{\mathbb{F},H,m}$ with soundness error $1/2$ and communication complexity $k^{O(1/r)}$. The verifier runs in time $k^{O(1/r)}$ and the prover runs in time $\mathsf{poly}(k)$.

Lemma 17 follows from Corollary 22 by applying Proposition 11, which is a generic transformation from any HIP, over the low degree extension encoding, into an IPP.

4.3 The Lower Bound

▶ **Lemma 23.** *Let $r = r(k) \geq 1$ be a constant. For every r-round IPP for Enc-MOD3, with respect to proximity parameter $\varepsilon = 1/10$, with query complexity q and communication complexity $c \geq \Omega(\log n)$, it holds that $\max(c, q) = n^{\Omega(1/r)}$.*

We remark that our proof of Lemma 23 gives a similar result even for super constant values of r (as long as $r = O\left(\sqrt{\frac{\log(n)}{\log\log(n)}}\right)$) but for simplicity we restrict ourselves to constant r. We also remark that the constants in the lemma's statement can be improved but we avoid optimizing them for sake of readability.

Proof. Throughout the proof of Lemma 23 all proofs of proximity refer to proximity parameter $\varepsilon = 1/10$.

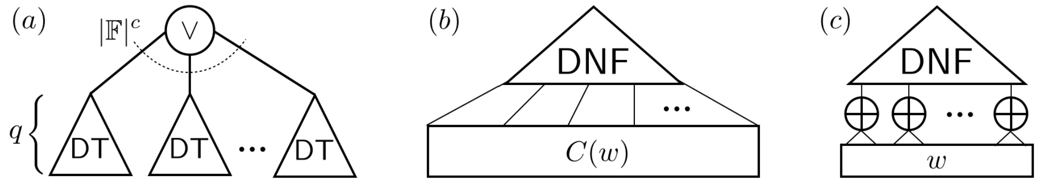


Figure 2 After fixing the randomness, an AMP for Π can be expressed as follows: (a) a disjunction over $O(2^c)$ decision trees of depth $q \cdot \log(|\mathbb{F}|)$, (b) a DNF formula with $O(2^{c+q\cdot\log(|\mathbb{F}|)})$ clauses of width $q \cdot \log|\mathbb{F}|$ over the linear code $C(w)$, and (c) a $\mathsf{DNF}_{\oplus}$ circuit of size $\tilde{O}(2^{c+q\cdot\log(|\mathbb{F}|)})$ over x.

The following proposition, due to [58] (building on [8, 39, 36]), shows that to prove Lemma 23, it suffices to prove a lower bound for AMPs, which are *public-coin* IPPs with only a *single* round of interaction between the verifier and prover. More precisely, in an AMP for a language $\mathcal{L}$, the verifier first sends a random string r to the prover, who responds with a proof π, which can depend on both the input x and the verifier's message r. Then, given π (and based on its original random coins r), the verifier needs to decide whether to accept or reject. (Note that the verifier is not allowed to toss additional coins after receiving the message from the prover.)

▶ **Proposition 24** (IPP to AMP). *If there exists an r-round (public or private coin)* IPP *for a language $\mathcal{L}$, with communication complexity $c \geq \log(n)$ and query complexity q, then there exists an* AMP *for $\mathcal{L}$ with communication complexity $c^{r+2} \cdot (\log(c) \cdot r)^{O(r)}$ and query complexity $c^{r+1} \cdot q \cdot (\log(c) \cdot r)^{O(r)}$.*

The proof of Proposition 24, which appears in [58, Section 4], proceeds by observing that the private-coin to public-coin transformation of [39] as well as the round reduction transformation of [8, 36], which are transformations on standard interactive proofs, can be applied to IPPs as well.

Thus, given Proposition 24, and using the fact that r is constant, to prove Lemma 23 it suffices to show that every AMP for Enc-MOD3 with query complexity q and communication complexity c satisfies $\max(c, q) = n^{\Omega(1)}$, or equivalently, since $n = O(k^3)$, that $\max(c, q) = k^{\Omega(1)}$. The following proposition, which is inspired by the [58] lower bound, shows that AMPs for properties of *linear codes* can be viewed as distributions over (relatively) small DNF*s of parities*. By DNF of parities, we refer to depth 3 circuits whose bottom layer consists of parity gates, middle layer consists of AND gates and top layer is a single OR gate. In the following we denote such circuits by $\mathsf{DNF}_{\oplus}$.

▶ **Proposition 25.** *Let $\mathbb{F}$ be an extension field of* GF(2)*, let $C : \mathbb{F}^k \to \mathbb{F}^n$ be an $\mathbb{F}$-linear code, and let $f : \{0,1\}^k \to \{0,1\}$. If there exists an* AMP *for $\Pi_f \overset{\text{def}}{=} \{C(x) : x \in \{0,1\}^k \wedge f(x) = 1\}$ with communication complexity $c \geq \log(n)$ and query complexity q, then there exists a distribution $\mathcal{D}$ over* $\mathsf{DNF}_{\oplus}$ *circuits of size $2^{O(c+q\cdot\log_2(|\mathbb{F}|))}$ such that $\Pr_{\varphi\sim\mathcal{D}}[\varphi(x) = f(x)] \geq 0.9$, for all $x \in \{0,1\}^k$.*

Proof. Let $\mathcal{V}$ be an AMP verifier for Π. We assume without loss of generality that $\mathcal{V}$ has soundness error at most 0.1 (e.g., by repeating the protocol in parallel $O(1)$ times). Recall that in an AMP protocol, for a given input $y \in \mathbb{F}^n$, the verifier first sends a random string r, then the prover replies with an alleged proof $\pi = \pi(r, y)$, and finally the verifier makes queries to y and decides whether to accept or reject. Denote by $\mathcal{V}^y_{r,\pi}$ the output of the verifier for a fixed string r, given oracle access to y and direct access to π.

For a fixed string r and alleged proof π, the verifier $\mathcal{V}^{y}_{r,\pi}$ can be represented as an $|\mathbb{F}|$-ary decision tree of depth q (on input y), which we denote by $D_{r,\pi} : \{0,1\}^k \to \{0,1\}$. The completeness and soundness requirements of an AMP guarantee that for a fixed string r, the verifier accepts an input $C(x)$ if and only if there exists a string π such that $\mathcal{V}^{C(x)}_{r,\pi} = 1$. Thus, $\mathcal{V}^{C(x)}_{r,\pi} = \bigvee_{\pi \in \{0,1\}^{O(c)}} D_{r,\pi}\big(C(x)\big)$ (see Figure 2(a)). Observe that by viewing elements of $\mathbb{F}$ in their bit-representation and assigning a clause for each accepting leaf in the decision tree, each $D_{r,\pi}$ can be represented as a *binary* DNF formula with $|\mathbb{F}|^{O(q)}$ clauses of width $O(q \cdot \log|\mathbb{F}|)$. Merging the two consecutive layers of disjunctions, we obtain a binary DNF formula that on input $y \in \mathbb{F}^n$ computes $\mathcal{V}^{y}_{r,\pi}$ with $2^{O(c+q\cdot\log_2(|\mathbb{F}|))}$ clauses of width $O(q \cdot \log_2(|\mathbb{F}|))$ each (see Figure 2(b) for an illustration).

We next observe that every linear combination over the field $\mathbb{F}$, which is an extension field of $\mathsf{GF}(2)$, can be represented by $\log_2(|\mathbb{F}|)$ linear combinations over $\mathsf{GF}(2)$.[24] Thus, we can view the function $C : \mathbb{F}^k \to \mathbb{F}^n$, which is an $\mathbb{F}$-linear function, as a $\mathsf{GF}(2)$-linear function $C : \mathsf{GF}(2)^{k\cdot\log_2(|\mathbb{F}|)} \to \mathsf{GF}(2)^{n\cdot\log_2(|\mathbb{F}|)}$. Hence, for every random string r, there exists a $\mathsf{DNF}_{\oplus}$ circuit of size:

$$2^{O(c+q\cdot\log_2(|\mathbb{F}|))} \cdot q \cdot \log_2(|\mathbb{F}|) + n \cdot \log_2(|\mathbb{F}|) = 2^{O(c+q\cdot\log(|\mathbb{F}|))}$$

(which is constructed by composing the code C with the DNF $\bigvee_{\pi \in \{0,1\}^{O(c)}} D_{r,\pi}$) that on input $x \in \{0,1\}^k$ outputs 1 if and only if there exists a proof π that $\mathcal{V}$ would accept, given input $C(x)$.

Therefore, there exists a distribution $\mathcal{D}$ over $\mathsf{DNF}_{\oplus}$s of size $2^{O(c+q\cdot\log(|\mathbb{F}|))}$ such that for every $x \in \{0,1\}^k$, it holds that $\Pr_{\varphi\in\mathcal{D}}[\varphi(x) = f(x)] \geq 0.9$. This concludes the proof of Proposition 25. ◀

Let $f_{\mathsf{MOD3}} : \{0,1\}^k \to \{0,1\}$ such that $f_{\mathsf{MOD3}} = 1$ if and only if $\sum_{i\in[k]} x_i \equiv 0 \pmod 3$. By Proposition 25, choosing $\Pi = \mathsf{Enc\text{-}MOD3}$, $C = \mathsf{LDE}^{\mathbb{F}}_{H,m}$, $f = f_{\mathsf{MOD3}}$, and using the (easy direction of) Yao's minimax principle, it suffices to show that there exists a distribution $\mathcal{X}$ over inputs in $\{0,1\}^k$ such that for every $\mathsf{DNF}_{\oplus}$ φ of size $\big(2^{O(c+q\cdot\log_2(|\mathbb{F}|))}\big)$ it holds that $\Pr_{x\in\mathcal{X}}[\varphi(x) = f_{\mathsf{MOD3}}(x)] < 0.9$ (where recall that $|\mathbb{F}| = \mathsf{polylog}(k)$). To that end, we shall use the celebrated result of Razborov [55] and Smolensky [62].

▶ **Theorem 26** (Razborov-Smolensky (see also [68, Theorem 2])). *Every $\mathsf{AC}^0(\oplus)$ circuit φ of size s and depth d satisfies*

$$\Pr_{x\in\{0,1\}^k}[\varphi(x) = f_{\mathsf{MOD3}}(x)] < \frac{2}{3} + O\left(\frac{\log(s)^d}{\sqrt{k}}\right).$$

This concludes the proof of Lemma 23. ◀

5 Implications for Classical Interactive Proofs

In this section, we derive from our hierarchy theorem implications to standard interactive proofs (in which the verifier can run in polynomial time). Loosely speaking, in Section 5.1 we show that the round reduction of public-coin interactive proofs, due to Babai and Moran [8], is (almost) optimal among all blackbox transformations, and in Section 5.2 we show that any proof that $\#\mathcal{P} \subseteq \mathcal{AM}$ will require using non-algebrizing techniques.

[24] Fix a linear combination $\alpha \in \mathbb{F}^t$ over $\mathbb{F}$ (the extension field). For every $i \in [\log_2(|\mathbb{F}|)]$, the function $\ell_{\alpha,i}(x) = \mathsf{bit}_i(\langle\alpha, x\rangle)$ that outputs the t^{th} bit of $\langle\alpha, x\rangle$ is a linear function over $\mathsf{GF}(2)$.

5.1 Blackbox Round Reduction Transformations

Babai and Moran [8] proved a "speedup" theorem, which loosely speaking, shows that very r-round public-coin interactive proof protocol can be transformed into an $(r-1)$-round protocol at the cost of increasing the communication complexity quadratically (some quantitative improvements were later obtained by Goldreich, Vadhan and Wigderson [36]). Combined with the private-coin to public-coin transformation of Goldwasser and Sipser [39], one can obtain a similar "speedup" theorem for private-coin interactive proofs.

Vadhan [66] considered the affect of certain transformations on interactive proofs. He introduced the notion of a "blackbox transformation" (defined below) and showed that the aforementioned private-coin to public-coin transformation, and a transformation from 2-sided error to 1-sided error of Goldreich, Mansour and Sipser [32], are (in a certain sense) optimal amongst all *black-box* transformation.

In this section, we use our hierarchy theorem to derive a similar result for the round reduction theorem of Babai and Moran. Following [66], we define a black-box transformation on interactive proofs as a procedure that takes as input an interactive proof $(\mathcal{P}, \mathcal{V})$ for some language $\mathcal{L}$ and outputs a new interactive proof $(\mathcal{P}', \mathcal{V}')$, for the same language $\mathcal{L}$, such that:

- The strategy of the verifier $\mathcal{V}'$ can be implemented by an algorithm given oracle access to the strategy of $\mathcal{V}$.
- The strategy of the prover $\mathcal{P}'$ can be implemented by a algorithm given oracle access to the strategy of both $\mathcal{P}$ and $\mathcal{V}$.

Here, the strategy of a party (i.e., prover or verifier) is the function that takes the party's random coins and the history of messages exchanged and outputs its next message. We stress that the new strategies $(\mathcal{P}', \mathcal{V}')$ cannot even explicitly look at the input x; their only access to the input x is given by queries to the strategies $(\mathcal{P}, \mathcal{V})$.

An r-to-r' blackbox round reduction transformation, for $r' < r$, is a black-box transformation that, given as input an r-round interactive proof, produces an r'-interactive proof (for the same language). We remark that the [8] round-reduction is a blackbox round reduction transformation, and we show that it is nearly optimal, out of all blackbox reductions.

▶ **Theorem 27.** *There exists a language $\mathcal{L}$ such that for every constant $r \geq 1$, there exists an r-round (public-coin) interactive proof $(\mathcal{P}, V)$ for $\mathcal{L}$, with communication complexity $c = c(n)$, such that for every r-to-r' blackbox round reduction transformation T, in the resulting interactive proof $(\mathcal{P}', \mathcal{V}') = T(\mathcal{P}, \mathcal{V})$ it holds that either the communication is at least $c^{\Omega(\sqrt{r}/r')}$ or $\mathcal{V}'$ invokes V at least $c^{\Omega(\sqrt{r}/r')}$ times.*

Proof. Let $r \in \mathbb{N}$ be a constant, and consider the language

$$\mathcal{L}_{\mathsf{MOD3}} = \left\{x \in \{0,1\}^k \ : \ \mathsf{wt}(x) = 0 \pmod 3\right\}_{k\in\mathbb{N}} .$$

Fix input length $k \in \mathbb{N}$, field $\mathbb{F}$, subset $H \subset \mathbb{F}$, and dimension $m = \frac{\log(k)}{\log\log(k)}$ such that $|H| = \log(k)$ and $|\mathbb{F}| = \Theta(|H|^2 m)$.

By Corollary 22, there exists an r-round HIP for $\mathcal{L}_{\mathsf{MOD3}}$, with respect to the code $\mathsf{LDE}_{\mathbb{F},H,m}$, with communication complexity $c \overset{\text{def}}{=} k^{O(1/\sqrt{r})}$. As noted in Proposition 9, this HIP implies an interactive proof $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}_{\mathsf{MOD3}}$, with communication complexity c. Recall that on input $x \in \{0,1\}^k$, the parties $(\mathcal{P}, \mathcal{V})$ invoke the HIP for $\mathcal{L}_{\mathsf{MOD3}}$, and the verifier checks the HIP's output claim by computing a single point of $\mathsf{LDE}_{\mathbb{F},H,m}(x)$.

Let T be an r-to-r' blackbox round reduction transformation on interactive proofs, and let $(\mathcal{P}', \mathcal{V}') = T(\mathcal{P}, \mathcal{V})$ be the resulting r'-round interactive proof for $\mathcal{L}_{\mathsf{MOD3}}$. Using $(\mathcal{P}', \mathcal{V}')$, we construct an r'-round ε-IPP for the language

$$\mathsf{Enc\text{-}MOD3} = \{\mathsf{LDE}_{\mathbb{F},H,m}(x) \ : \ x \in \mathcal{L}_{\mathsf{MOD3}}\}\,.$$

Recall that $\mathcal{V}$ only computes $\mathsf{LDE}_{\mathbb{F},H,m}(x)$ and queries it at a single point, and so each oracle call to $\mathcal{V}$ that $\mathcal{V}'$ makes can be emulated by making a single query to $\mathsf{LDE}_{\mathbb{F},H,m}(x)$. Therefore, we can view $(\mathcal{P}', \mathcal{V}')$ as an HIP, with respect to $\mathsf{LDE}_{\mathbb{F},H,m}$, for $\mathcal{L}_{\mathsf{MOD3}}$, with communication complexity c.

By applying Proposition 11 on $(\mathcal{P}', \mathcal{V}')$, we obtain an r'-round IPP for $\mathsf{Enc\text{-}MOD3}$; denote its communication complexity by C and query complexity by Q. Finally, by Lemma 23 we have that:

$$\max(C, Q) = k^{\Omega(1/r')} = c^{\Omega(\sqrt{r}/r')}. \qquad \blacktriangleleft$$

5.2 The Algebrization Barrier

The *relatization* framework, introduced by Baker, Gill, and Solovay [9], tried to capture the intuition that we not understand how circuits operate and therefore we may as well treat them as black-boxes. Later on, the seminal result of [52, 60] showed that even without understanding how circuits operate, we can still do more than just evaluate them (i.e., treat them as oracles). Specifically, arithmetizing the circuit, allows us to evaluate points in a *low degree extension* of the function computed by the circuit. The latter cannot be done only via oracle access and has turned out to be incredibly useful.

The *algebrization* framework, introduced by Aaronson and Wigderson [1], tries to capture this additional power. Specifically, in this framework, rather than just giving oracle access to the given function, we give oracle access also to a low degree extension of the function. Results such as $\mathsf{IP} = \mathsf{PSPACE}$ can be showed to have "algebrizing" proofs. Despite the power that we obtain by having access to the low degree extension of the function, [1] also showed that some central questions in complexity theory cannot be proved within this framework (i.e., by "algebrizing") techniques.

Loosely speaking, for two complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the inclusion $\mathcal{C}_1 \subseteq \mathcal{C}_2$ is said to algebrize if $\mathcal{C}_1^A \subseteq \mathcal{C}_2^{\tilde{A}}$ for every oracle A and every low-degree extension $\tilde{A}$ of A. (See [1] for the precise definition, discussions and many more details.) We say that proving the inclusion $\mathcal{C}_1 \subseteq \mathcal{C}_2$ requires non-algebrizing techniques, or cannot be proved via algebrizing techniques, if the inclusion does *not* algebrize.

Before stating our results, we point out that there is an intimate connection (or a high level equivalence) between the class the algebrized class $\mathsf{IP}^{\tilde{A}}$ (where $\tilde{A}$ is the low degree extension of some oracle A) and the notion of HIPs (with respect to the low degree extension encoding). Indeed, in both cases the verifier needs to verify a property of some string, given oracle access to its low degree extension and interaction with the prover. For an $\mathsf{IP}^{\tilde{A}}$ the string is the truth table of A and for HIPs the string is simply the input.

Using this relation, we use our hierarchy theorem to show that the inclusion $\#\mathcal{P} \subseteq \mathcal{AM}$, which is widely believed *not* to hold[25], cannot be proved via algebrizing techniques. As a matter of fact, the proof of Theorem 28 can be easily extended to show that even the containment of $\#\mathcal{P}$ in a powerful variant of $\mathcal{AM}$ in which, for inputs of length N, the verifier is allowed to run in time $2^{o(N)}$ and with $2^{o(N)}$ communication, cannot be proved via algebrizing techniques.

[25] In particular it implies the collapse of the polynomial hierarchy to its second level.

▶ **Theorem 28.** *There exists an oracle A and a low-degree extension $\tilde{A}$ of A such that $\#\mathcal{P}^A \not\subseteq \mathcal{AM}^{\tilde{A}}$.*

Proof Sketch. Consider the problem #CSAT, which is the problem of counting the number of satisfying assignments of a given (Boolean) circuit C, and recall that #CSAT is $\#\mathcal{P}$-complete. Let $A : \{0,1\}^N \to \{0,1\}$ be an oracle and consider an input circuit C that, given as input $x \in \{0,1\}^N$, just outputs $A(x)$. We associate A with its truth table, which is a string of length 2^N. Let $\tilde{A} = \mathsf{LDE}_{\mathbb{F},H,m}(A)$, where $\mathbb{F}, H, m$ are defined as in Section 4.1, with respect to the parameter $k = 2^N$.

Observe that if $\#\mathcal{P}^A \subset \mathcal{AM}^{\tilde{A}}$, then there exists an $\mathcal{AM}$ proof system for computing the number of satisfying assignments of the circuit C, which is exactly the Hamming weight of A (viewed as an k-bit string), in which the communication complexity is $\mathsf{poly}(N)$ and in which the verifier only makes $\mathsf{poly}(N)$ oracle queries to $\tilde{A}$. Thus, following Proposition 11, we can obtain from this $\mathcal{AM}$ proof system a 1-round IPP for Enc-MOD3 with communication and query complexities $\mathsf{poly}(N) = \mathsf{polylog}(k)$, which violates the lower bound in Lemma 23. ◀

▶ Remark (Using Prime Order Fields). We remark that the proof of Theorem 28 is strongly based on the fact that we take a low degree extension over a field that has characteristic 2. Our result can extend to other constant size characteristics but we do not know how to extend it to arbitrary fields. In fact, it is consistent with our result (however unlikely) that there is a proof that $\#\mathcal{P} \subseteq \mathcal{AM}$ based (in a crucial way) on taking the low degree extension of the circuit with respect to a large prime order field.

We remark that we are unaware of any complexity class containments in the literature that are only known based on algebrization using *prime* order fields.[26]

Acknowledgements. We are grateful to Guy Rothblum for sharing [45] with us. We thank Oded Goldreich for his encouragement and support, for many technical and conceptual discussions on the contents of this work, and for valuable comments on its presentation. We thank Justin Holmgren for the proof of Proposition 33. Finally, we wish to thank Shafi Goldwasser, Guy Rothblum, and Rocco Servedio for insightful conversations.

References

1 Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1:2:1–2:54, February 2009. doi:10.1145/1490270.1490272.

2 William Aiello, Shafi Goldwasser, and Johan Håstad. On the power of interaction. *Combinatorica*, 10(1):3–25, 1990. doi:10.1007/BF02122692.

3 Miklós Ajtai, János Komlós, and Endre Szemerédi. An O(n log n) sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 1–9, 1983. doi:10.1145/800061.808726.

4 Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. doi:10.1007/s00493-003-0025-0.

5 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991. doi:10.1145/103418.103428.

[26] The original proof of the IP = PSPACE theorem by Shamir [60] does use prime order fields in an important way, however, the more recent proof of the same result by Goldwasser *et al.* [37] can be based on fields of arbitrary characteristic (see also [53] that gives a proof based on general tensor codes).

6 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056.

7 Laszlo Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Washington, DC, USA, 1986. IEEE Computer Society. doi:10.1109/SFCS.1986.15.

8 László Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

9 Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P=?NP question. *SIAM Journal on computing*, 4(4):431–442, 1975.

10 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

11 Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 687–706. SIAM, 2014.

12 Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 222–234, Berlin, Heidelberg, 2009. Springer-Verlag. doi:10.1007/978-3-642-02927-1_20.

13 Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur–Merlin communication. In *30th Conference on Computational Complexity (CCC 2015)*, volume 33, pages 217–243. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

14 Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

15 Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 185–202, 2013. doi:10.1007/978-3-642-40084-1_11.

16 Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

17 Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

18 Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. Streaming verification in data analysis. *arXiv preprint arXiv:1509.05514*, 2015.

19 Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004. doi:10.1016/j.ic.2003.09.005.

20 Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 483–500, 2014. doi:10.1145/2554797.2554841.

21 Eldar Fischer, Oded Lachish, and Yadu Vasudev. Trading query complexity for sample-based testing and multi-testing scalability. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1163–1182. IEEE, 2015.

22 Lance Fortnow and Michael Sipser. Are there interactive protocols for CO-NP languages? *Inf. Process. Lett.*, 28(5):249–251, 1988. `doi:10.1016/0020-0190(88)90199-8`.

23 Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992. `doi:10.1016/0020-0190(92)90195-2`.

24 Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

25 Oded Goldreich. Valiant's polynomial-size monotone formula for majority. Unpublished, 2011. URL: `http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-maj.pdf`.

26 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. URL: `http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html`.

27 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

28 Oded Goldreich and Tom Gur. Universal locally testable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:42, 2016. URL: `http://eccc.hpi-web.de/report/2016/042`.

29 Oded Goldreich and Tom Gur. Universal locally verifiable codes and 3-round interactive proofs of proximity for CSP. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:192, 2016. URL: `http://eccc.hpi-web.de/report/2016/192`.

30 Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 1–41, 2015. `doi:10.4230/LIPIcs.CCC.2015.1`.

31 Oded Goldreich, Tom Gur, and Ron D. Rothblum. Proofs of proximity for context-free languages and read-once branching programs - (extended abstract). In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 666–677, 2015. `doi:10.1007/978-3-662-47672-7_54`.

32 Oded Goldreich, Yishay Mansour, and Michael Sipser. Interactive proof systems: Provers that never fail and random selection (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 449–461, 1987. `doi:10.1109/SFCS.1987.35`.

33 Oded Goldreich and Dana Ron. On sample-based testers. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:109, 2013. URL: `http://eccc.hpi-web.de/report/2013/109`.

34 Oded Goldreich and Dana Ron. On learning and testing dynamic environments. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:29, 2014. URL: `http://eccc.hpi-web.de/report/2014/029/`.

35 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006. `doi:10.1145/1162349.1162351`.

36 Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002. `doi:10.1007/s00037-002-0169-0`.

37 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

38 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. `doi:10.1137/0218012`.

39 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

40 Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:49, 2015. URL: `http://eccc.hpi-web.de/report/2015/049`.

41 Mika Göös, Toniann Pitassi, and Thomas Watson. Zero-information protocols and unambiguity in Arthur-Merlin communication. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 113–122, 2015. `doi:10.1145/2688073.2688074`.

42 Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. *Information and Computation*, 243:145–165, 2015. `doi:10.1016/j.ic.2014.12.011`.

43 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Computational Complexity*, pages 1–109, 2016. `doi:10.1007/s00037-016-0136-9`.

44 Shlomo Hoory, Avner Magen, and Toniann Pitassi. Monotone circuits for the majority function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 410–425, 2006. `doi:10.1007/11830924_38`.

45 Yael Kalai and Guy N. Rothblum. Constant-round interactive proofs for NC^1. Unpublished observation, 2009.

46 Yael Tauman Kalai and Ran Raz. Interactive PCP. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 536–547, 2008. `doi:10.1007/978-3-540-70583-3_44`.

47 Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574, 2013. `doi:10.1145/2488608.2488679`.

48 Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014. `doi:10.1145/2591796.2591809`.

49 Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity - [extended abstract]. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 422–442, 2015. `doi:10.1007/978-3-662-48000-7_21`.

50 Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

51 Hartmut Klauck. On Arthur Merlin games in communication complexity. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on*, pages 189–199. IEEE, 2011.

52 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. `doi:10.1145/146585.146605`.

53 Or Meir. IP = PSPACE using error-correcting codes. *SIAM J. Comput.*, 42(1):380–403, 2013. `doi:10.1137/110829660`.

54 Ran Raz, Gábor Tardos, Oleg Verbitsky, and Nikolai Vereshagin. Arthur-Merlin games in boolean decision trees. In *Computational Complexity, 1998. Proceedings. Thirteenth Annual IEEE Conference on*, pages 58–67. IEEE, 1998.

55 A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. Notes of the Academy of Science of the USSR: 41(4) : 333-338, 1987.

56 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016. doi:10.1145/2897518.2897652.

57 Guy N. Rothblum. *Delegating computation reliably: paradigms and constructions*. PhD thesis, Massachusetts Institute of Technology, 2009.

58 Guy N. Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: Delegating computation in sublinear time. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, 2013.

59 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

60 Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

61 Alexander A Sherstov. The multiparty communication complexity of set disjointness. In *Proceedings of the 44th symposium on Theory of Computing*, pages 525–548. ACM, 2012.

62 R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 77–82, New York, NY, USA, 1987. ACM. doi:10.1145/28395.28404.

63 Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1992. UMI Order No. GAX93-30747.

64 Madhu Sudan. *Efficient Checking of Polynomials and Proofs anf the Hardness of Approximation Problems*, volume 1001 of *Lecture Notes in Computer Science*. Springer, 1995. doi:10.1007/3-540-60615-7.

65 Justin Thaler. Semi-streaming algorithms for annotated graph streams. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 17:1–17:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.17.

66 Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 200–207, 2000. doi:10.1145/335305.335330.

67 Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.

68 Emanuele Viola. Guest column: correlation bounds for polynomials over $\{0\ 1\}$. *SIGACT News*, 40(1):27–44, 2009. doi:10.1145/1515698.1515709.

69 Richard Ryan Williams. Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 2:1–2:17, 2016. doi:10.4230/LIPIcs.CCC.2016.2.

A Talmudic Discussions

A.1 Why GKR and not other Interactive Proofs?

One may wonder whether we could base our upper bound on other interactive proofs from the literature. Other than the protocols of [37, 45], two other general purpose interactive proof-systems that come to mind are Shamir's[27][60] protocol for $\mathsf{IP} = \mathsf{PSPACE}$ and a recent protocol of Reingold, Rothblum and Rothblum [56] that gives constant-round interactive proofs for bounded-space computations.

[27] Indeed, here we specifically refer to Shamir's [60] protocol and not to the [52] protocol (on which [60] builds).

Shamir's protocol is not suitable for our needs both because it is not constant-round, and, perhaps more fundamentally, because the verifier in Shamir's protocol needs to access the low-degree extension of the input over a field that is only determined during the interaction (recall that the verifier in Shamir's protocol chooses a *random* prime p, and the players both work over the field of integers modulo p). For our purposes the field has to be fixed a priori (since we want the input for the IPP to be encoded under the LDE code corresponding to that field).

As for the protocol of [56], the latter does actually yield a constant-round HIP for $\mathcal{L}_{\mathsf{MOD3}}$ (which can be modified to yield an IPP for Enc-MOD3 as above) but the tradeoff that it offers between rounds and the verifier's complexity is exponentially worse than what we obtain. More specifically, for every constant $r \geq 1$, the [56] protocol yields a $2^{\tilde{O}(r)}$-round HIP for $\mathcal{L}_{\mathsf{MOD3}}$ with verification time roughly $2^{\tilde{O}(r)} \cdot k^{1/r}$. In contrast, we obtain an $O(r^2)$-round HIP with verification time roughly $\mathsf{poly}(r) \cdot k^{1/r}$.

A.2 An Alternative Candidate Language for the Round Hierarchy Theorem

The language for which we proved our round hierarchy consists of encodings of strings whose Hamming weight is divisible by 3. As described next, it seems as though a similar result can be obtained for a related language Enc-Maj that consists of encodings of strings $x \in \{0,1\}^k$ with $\mathsf{wt}(x) \geq k/2$, although there are some technical difficulties to overcome.

First note that the lower bound for Enc-Maj follows along the same lines as our lower bound for Enc-MOD3, where now we use the fact that $\mathsf{AC}_0[2]$ circuits cannot approximate the majority function [55, 62]. In contrast, showing an upper bound (i.e., an IPP or HIP) introduces some new difficulties. As explained in Section 1 (and formalized in Section 4), our upper bound for Enc-MOD3 is based on the observation that computing the sum, modulo 3, of the bits of an input string can be done by a (highly uniform) NC^1 circuit. Given this observation, we based our protocol on a variant of the GKR interactive proof for small-depth computations.

For Enc-Maj, we could similarly hope to base our protocol on an NC^1 circuit, but this time we need a circuit that computes the majority function. Obtaining such a circuit is less trivial and here we encounter some difficulties:

- Valiant [67] (see the presentation of Goldreich [25]) gave a *non-uniform* construction of an NC^1 circuit for majority. We could base our protocol on this result and obtain a *non-uniform* verifier (and in particular, its running time would be super-linear, although it would still have the desired query and communication complexities).
- The aforementioned construction of [67] can actually be shown to produce a (highly-uniform) *randomized* construction. That is, there exists a randomized logspace Turing machine that given as input 1^n, with all but exponentially vanishing probability, produces an NC^1 circuit, on n-bit strings, that computes the majority function correctly (on all inputs). We could have our verifier run this procedure to obtain the desired NC^1 circuit, but this would introduce an (exponentially small) completeness error, which we would like to avoid.
- Lastly, we mention that the celebrated [3] sorting network of Ajtai, Komlós and Szemerédi gives rise to a uniform (and deterministic) construction of an NC^1 circuit for majority (by sorting the input bits and outputting the median). This construction is quite complex and in particular we have not verifed whether it satisfies the uniformity condition that is required for the [45] result.[28]

[28] We note that other *partial*, but arguably simpler, de-randomization results of Valiant's formula have

B From HIPs to IPPs (Proof of Proposition 11)

The two main ingredients that we shall use to prove Proposition 11 are the well-known low (individual) degree test[29] for multivariate polynomials [59, 64, 4], and the self-correction procedure for polynomials [23, 64].

▶ **Lemma 29** (Individual Degree Test)**.** *Let $d, m \in \mathbb{N}$ such that $dm < |\mathbb{F}|/10$ and $\varepsilon \in (0, 1/10)$. Denote by $\mathsf{Poly}_{d,m,\mathbb{F}}$ the set of all m-variate, individual degree d polynomials over $\mathbb{F}$. Then, there exists an ε-tester for $\mathsf{Poly}_{d,m,\mathbb{F}}$ with query complexity $dm \cdot \mathsf{poly}(1/\varepsilon)$.*

▶ **Lemma 30.** *Let $\varepsilon < 1/3$ and $d, m \in \mathbb{N}$ such that $d \leq |\mathbb{F}|$. There exists an algorithm (corrector) that, given $x \in \mathbb{F}^m$ and oracle access to an m-variate function $f : \mathbb{F}^m \to \mathbb{F}$ that is ε-close to a polynomial p of individual degree d, makes $O(d \cdot m)$ queries and outputs $p(x)$ with probability $9/10$. Furthermore, if f has total degree d, the algorithm outputs $p(x)$ with probability 1.*

Given Lemmas 29 to 30, we can now describe the IPP (with respect to some proximity parameter ε) for $\mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L})$. Recall that the verifier is given oracle access to a function $f : \mathbb{F}^m \to \mathbb{F}$ and the prover is given direct access to f. Assume, without loss of generality, that the HIP for $\mathcal{L}$ has soundness error $1/10$.[30]

First, the verifier and prover run the HIP protocol for $\mathcal{L}$ with respect to the input $f|_{H^m}$. (Recall that an HIP does not even query its input and therefore, so far, no queries have been made.) If the HIP verifier rejects then we immediately reject. Otherwise, the verifier outputs a pair $(z, \nu) \in \mathbb{F}^m \times \mathbb{F}$ (with the associated claim that $f(z) = \nu$). Then, the verifier runs the individual degree tester of Lemma 29 on f, with respect to proximity parameter ε, individual degree $|H| - 1$ (and soundness error $1/3$). If the low degree test rejects, the verifier immediately rejects. Lastly, the verifier decodes f at point z, using the self-correction procedure of Lemma 30, again with soundness error $1/10$. The procedure outputs a value ν'. The verifier accepts if $\nu = \nu'$ and otherwise it rejects.

Completeness follows from the perfect completeness of the HIP, the low degree test and the local self-correction. For soundness, let $f : \mathbb{F}^m \to \mathbb{F}$ be a function such that f is ε-far from $\mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L})$ and fix a cheating prover strategy $\mathcal{P}^*$. Consider first the case that f is ε-far from an individual degree $|H| - 1$ polynomial. In this case, by the low degree test, with probability at least $2/3$, the verifier rejects and we are done. Thus, we can assume that f is ε-close to some individual degree $|H| - 1$ polynomial $P : \mathbb{F}^m \to \mathbb{F}$. Observe that since f is ε-far from $\mathsf{LDE}_{\mathbb{F},H,m}(\mathcal{L})$ it must be the case that $P|_{H^m} \notin \mathcal{L}$.

We view the HIP as being applied to $P|_{H^m}$. By the soundness of the HIP, when the verifier interacts with any cheating prover (and in particular $\mathcal{P}^*$) with probability $9/10$ it either rejects (in which case we also reject) or it outputs a pair $(z, \nu) \in \mathbb{F}^m \times \mathbb{F}$ such that $P(z) \neq \nu$. The verifier reads the point z with self-correction and so, with probability at least $9/10$ it will obtain the actual value $\nu' = P(z)$ and reject when comparing ν' and ν. Thus, with probability $0.9^2 \geq 2/3$ our verifier rejects.

been obtained by [44] and [15]. However, these partial derandomizations do not seem to suffice for our purposes.

[29] Actually, the cited works provide a test for *total* degree. A test for *individual* degree (which is implicit in [35, Section 5.4.2]) can be obtained via a simple reduction (see, e.g., [43, Theorem A.8]).

[30] Indeed, parallel repetition of IPPs decreases their soundness error at an exponential rate (see [31, Appendix A] for details).

C The Sumcheck Protocol (Proof of Lemma 10)

In this appendix we prove Lemma 10.

We use a variant of the sumcheck protocol that takes r rounds, where for simplicity we assume that r divides m. We maintain the invariant that before the i^{th} rounds begins, both the verifier and the prover agree on values $w_1, \ldots, w_{i-1} \in \mathbb{F}^{m/r}$ and $\nu_{i-1} \in \mathbb{F}$, where $\nu_0 \stackrel{\text{def}}{=} 0$. For every $i \in [r]$, the i^{th} round of the sumcheck protocol is as follows.

1. The prover sends to the verifier the individual degree $|H|-1$ polynomial $P_i : \mathbb{F}^{m/r} \to \mathbb{F}$ (by specifying its coefficients), defined as:
$$P_i(z) \stackrel{\text{def}}{=} \sum_{x_{i+1},\ldots,x_r \in H^{m/r}} P(w_1, \ldots, w_{i-1}, z, x_{i+1}, \ldots, x_r).$$
2. The verifier receives a polynomial $Q_i : \mathbb{F}^{m/r} \to \mathbb{F}$ (which is allegedly equal to P_i) and checks that $\sum_{z \in H^{m/r}} Q_i(z) = \nu_{i-1}$.
3. The verifier select uniformly at random $w_i \in \mathbb{F}^{m/r}$ and sends w_i to the prover.
4. Set $\nu_i \stackrel{\text{def}}{=} Q_i(w_i)$.

At the end of the protocol, the verifier outputs $((w_1, \ldots, w_r), \nu_r) \in \mathbb{F}^m \times \mathbb{F}$.

The running times and communication complexity of the protocol can be readily verified. We proceed to show that completeness and soundness hold.

C.1 Completeness

Let $P : \mathbb{F}^m \to \mathbb{F}$ be an individual degree $|H|-1$ polynomial such that $\sum_{x \in H^m} P(x) = 0$. In this case, at every round $i \in [\rho]$, the prover sends the polynomial $Q_i \equiv P_i$. Hence, for every $i \in [r]$:

$$\begin{aligned}
\sum_{z \in H^{m/r}} Q_i(z) &= \sum_{z \in H^{m/r}} P_i(z) \\
&= \sum_{z \in H^{m/r}} \sum_{x_{i+1},\ldots,x_r \in H^{m/r}} P(w_1, \ldots, w_{i-1}, z, x_{i+1}, \ldots, x_r) \\
&= P_{i-1}(w_{i-1}) \\
&= Q_{i-1}(w_{i-1}) \\
&= \nu_{i-1}
\end{aligned}$$

and so all of the verifier's checks pass. At the end of the protocol the verifier outputs $((w_1, \ldots, w_r), \nu_r) \in \mathbb{F}^m \times \mathbb{F}$ and $\nu_r = P_r(w_r) = P(w_1, \ldots, w_r))$ as required.

C.2 Soundness

Let $P : \mathbb{F}^m \to \mathbb{F}$ be an individual degree $|H|-1$ polynomial such that $\sum_{x \in H^m} P(x) \neq 0$ and fix a cheating prover strategy $\mathcal{P}^*$.

The next two claims relate the polynomials Q_i sent by the prover to the corresponding polynomials P_i (recall that P_i was defined as $P_i(z) = \sum_{x_{i+1},\ldots,x_r \in H^{m/r}} P(w_1, \ldots, w_{i-1}, z, x_{i+1}, \ldots, x_r)$). Recall that both polynomials depend only on $w_1, \ldots, w_{i-1}$.

▶ **Claim 31.** *If $Q_1 \equiv P_1$, then the verifier rejects with probability* 1.

Proof. Observe that $\sum_{x_1 \in H^{m/r}} P_1(x_1) = \sum_{z \in H^m} P(z) \neq 0$, and so, if $Q_1 \equiv P_1$, then the verifier rejects when testing that $\sum_{z \in H^{m/r}} Q_1(z) = \nu_0 = 0$. ◀

▶ **Claim 32.** *For every $i \in [r-1]$ and every $w_1, \dots, w_{i-1} \in \mathbb{F}^{m/r}$, if $Q_i \not\equiv P_i$ then, with probability $1 - \frac{(m/r)\cdot|H|}{|\mathbb{F}|}$ over the choice of w_i, if $Q_{i+1} \equiv P_{i+1}$ then the verifier rejects.*

Proof. Since the (total degree $(m/r)\cdot(|H|-1)$) polynomials Q_i and P_i differ, by the Shwartz-Zippel lemma (Lemma 5), with probability $1 - \frac{(m/r)\cdot|H|}{|\mathbb{F}|}$ over the choice of $w_i \in_R \mathbb{F}^{m/r}$, it holds that $Q_i(w_i) \neq P_i(w_i)$. If the latter event occurs and the prover sends $Q_{i+1} \equiv P_{i+1}$, then the verifier rejects when testing whether $\sum_{z \in H^{m/r}} Q_{i+1}(z) = \nu_i$, since

$$\nu_i = Q_i(w_i) \neq P_i(w_i) = \sum_{z \in H^{m/r}} P_{i+1}(z) = \sum_{z \in H^{m/r}} Q_{i+1}(z). \qquad ◀$$

By Claims 31 and 32 and an application of the union bound, with probability $1 - (r-1) \cdot \frac{(m/r)\cdot|H}{|\mathbb{F}|}$, if there exists an $i \in [r-1]$ such that $Q_i \not\equiv P_i$ but $Q_{i+1} \equiv P_{i+1}$ then the verifier rejects. However, by Claim 31, we can assume that $Q_1 \not\equiv P_1$ and so we get that with probability $1 - (r-1) \cdot \frac{(m/r)\cdot|H}{|\mathbb{F}|}$ either the verifier rejects or $Q_r \not\equiv P_r$. Note that if $Q_r \not\equiv P_r$ then by the Shwartz Zippel Lemma with probability $1 - \frac{(m/r)\cdot|H|}{|\mathbb{F}|}$ it holds that $Q_r(w_r) \neq P_r(w_r)$ and therefore:

$$\nu_r = Q_r(w_r) \neq P_r(w_r) = P(w_1, \dots, w_r)$$

and so the soundness condition holds, with soundness error $(r-1) \cdot \frac{(m/r)\cdot|H}{|\mathbb{F}|} + \frac{(m/r)\cdot|H}{|\mathbb{F}|} = \frac{m\cdot|H}{|\mathbb{F}|}$.

D Interactive Proof for Vanishing-Subcube (Proof of Proposition 14)

Let $\mathbb{F}$ be a constructible field ensemble, let $H \subseteq G \subseteq \mathbb{F}$ be ensembles of subsets, and let $m \in \mathbb{N}$. Recall that $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$ is the set of all functions $f : G^m \to \mathbb{F}$ that vanish on H^m (i.e., $f|_{H^m} \equiv 0$). We show that for every $r \in [m]$, there exists an $r+2$-round (public-coin) HIP for $\mathsf{Vanishing\text{-}Subcube}_{\mathbb{F},H,m,G}$, with respect to the code $\mathsf{LDE}_{\mathbb{F},G,m}$.

Recall that in an HIP with respect to the code $\mathsf{LDE}_{\mathbb{F},G,m}$, the input should be thought of as an m-variate polynomial P with individual degree $|G|-1$. The prover has direct access to P and the verifier needs to output a pair $(z, \nu) \in \mathbb{F}^m \times \mathbb{F}$, with the associated claim that $P(z) = \nu$.

For a given function $P : \mathbb{F}^m \to \mathbb{F}$, we define the polynomial $\tilde{P}(x) = \sum_{z \in H^m} \delta(z, x) \cdot P(z)$, where $\delta : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ is an individual degree $|H|-1$ polynomial such that for every $a, b \in H^m$, it holds that $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise (and δ is arbitrary in $\mathbb{F}^{2m} \backslash H^{2m}$).[31]

To check that P is identically 0 in H^m, the verifier first chooses at random $r \in \mathbb{F}^m$ and sends r to the prover. Now, the prover and verifier run an interactive proof to check that $\tilde{P}(r) = 0$, by invoking the sumcheck protocol with respect to the summation $\sum_{z \in H^m} \delta(z, r) \cdot P(z) = 0$, where we observe that the polynomial $\delta(\cdot, r) \cdot P(\cdot)$ has individual degree $|H| + |G| - 1$. If the sumcheck verifier rejects, then we immediately reject. Otherwise, the sumcheck verifier outputs a pair $(z, \nu) \in \mathbb{F}^m \times \mathbb{F}$, and the prover then sends the value $\nu' = P(z)$. Finally, the verifier checks that $\delta(z, r) \cdot \nu' = \nu$ and if so outputs (z, ν').

For completeness, note that if P is identically 0 in H^m, then $\tilde{P}$ is identically 0 in $\mathbb{F}^m$. In particular, with probability 1 over the choice of r it holds that $\tilde{P}(r) = \sum_{z \in H^m} \delta(z, r) \cdot P(h) = 0$. Thus, by the completeness of the sumcheck protocol, the sumcheck verifier outputs a pair

[31] We note that $\tilde{P}$ is in fact the low degree extension of the function P, when the latter is restricted to H^m.

(z,ν) such that $\delta(z,r)\cdot P(z)=0$. The prover now sends the value $\nu'=P(z)$, and so the verifier's check that $\delta(z,r)\cdot\nu'=\nu$ passes, and it outputs the claim (z,ν'), which is correct since $P(z)=\nu'$.

As for soundness, if P is not identically 0 in H^m, then by definition, $\tilde{P}$ is not identically 0 in $\mathbb{F}^m$, and therefore by the Schwartz-Zippel lemma (see Lemma 5), with probability $1-\frac{m\cdot(|H|-1)}{|\mathbb{F}|}$ over the choice of r, it holds that $\tilde{P}(r)\neq 0$. Thus, the sumcheck protocol is invoked on the sum $\sum_{z\in H^m}\delta(z,r)\cdot P(z)\neq 0$ and so, with probability $1-\frac{m\cdot(|H|+|G|-2)}{|\mathbb{F}|}$ either the sumcheck verifier rejects, or it outputs a claim (z,ν) such that $\delta(z,r)\cdot P(z)\neq\nu$. Assuming the latter happens, if the prover now sends $\nu'=P(z)$, then the verifier rejects. Hence, it must send $\nu'\neq P(z)$, and so the verifier outputs the incorrect claim (z,ν').

E Efficiently Computing $\widetilde{\mathsf{MOD3}}_t$

Recall that $\widetilde{\mathsf{MOD3}}_t:\mathbb{K}^t\to\mathbb{K}$ was defined as the (unique) individual degree 2 polynomial such that for every $h\in\{0,1,2\}^t$ it holds that $\widetilde{\mathsf{MOD3}}_t(h)=\sum_{i\in[t]}h_i \pmod 3$. In this section we show that $\widetilde{\mathsf{MOD3}}$ is efficiently computable. Namely, that given a point $z\in\mathbb{K}^t$, one can compute $\widetilde{\mathsf{MOD3}}_t(z)$ in time $\mathsf{poly}(t,\log(|\mathbb{K}|))$.

▶ **Proposition 33.** *Let $\mathbb{K}$ be a constructible field ensemble. There exists a $\mathsf{poly}(t,\log(|\mathbb{K}|))$-time algorithm that given a point $z\in\mathbb{K}^t$ outputs the value $\widetilde{\mathsf{MOD3}}_t(z)$.*

Proof. To prove Proposition 33, we first show that for every $\sigma\in\{0,1,2\}$ and $i\in[t]$, we can construct a size $\mathsf{poly}(i)$ uniform arithmetic circuit over $\mathbb{K}$ that computes the function $F_i^{(\sigma)}:\mathbb{K}^i\to\mathbb{K}$, which is defined as the unique individual degree 2 polynomial such that:

$$\forall h\in\{0,1,2\}^i,\quad F_i^{(\sigma)}(h)=\begin{cases}1 & \text{if } \sum_{i\in[t]}h_i=\sigma \pmod 3\\ 0 & \text{otherwise}\end{cases}.$$

where the summation is over integers modulo 3. Despite their similarity, note that $\widetilde{\mathsf{MOD3}}_t$ is the low degree extension of a function that *computes* the sum modulo 3 of its input, whereas $F_t^{(\sigma)}$ is the low degree extension of a function that *indicates* whether the sum modulo 3 is congruent to σ.

Given arithmetic circuits that compute $\mathbb{F}_t^{(\sigma)}$, we can now compute $\widetilde{\mathsf{MOD3}}_t:\mathbb{K}^t\to\mathbb{K}$ as:

$$\widetilde{\mathsf{MOD3}}_t(z)=\sum_{\sigma\in\{0,1,2\}}\sigma\cdot F_i^{(\sigma)}(z), \tag{10}$$

where here the arithmetic is over the field $\mathbb{K}$, and the equality follows from the fact that both sides of the equation are polynomials of individual degree 2 that agree on $\{0,1,2\}^t$ and therefore must agree on $\mathbb{K}^t$. Thus, it remains to prove the following claim.

▶ **Claim 34.** *For every $\sigma\in\{0,1,2\}$ and $i\in\mathbb{N}$, there exists an arithmetic circuit of size $O\left(i^{\log_2(6)}\right)$ over $\mathbb{K}$ that computes $F_i^{(\sigma)}$.*

Proof. We prove the proposition for i's that are powers of two and note that the general case follows easily (e.g., by using a circuit of size that is the nearest power of two and fixing some of its inputs to 0).

The proof is by induction on i, where the base case $i=1$, is trivial. Fix i (that is a power of two) and suppose that we have constructed arithmetic circuits for computing $F_i^{(\sigma)}$ for every $\sigma\in\{0,1,2\}$.

Fix $\tau \in \{0,1,2\}$. The main observation is that for every $z_1, z_2 \in \mathbb{K}^i$ it holds that

$$F_{2i}^{(\tau)}(z_1,z_2) = \sum_{\sigma \in \{0,1,2\}} F_i^{(\sigma)}(z_1) \cdot F_i^{(\tau-\sigma \bmod 3)}(z_2), \tag{11}$$

where the equality follows from the fact that that both sides of the equation are polynomials of individual degree 2 that agree on $\{0,1,2\}^i$ and therefore must agree on $\mathbb{K}^{2i}$.

Denoting by S_i the size of the arithmetic circuit that Equation (11) yields for $F_i^{(\sigma)}$, it holds that:

$$S_{2i} = 6 \cdot S_i + c = \cdots = 6^{\log(2i)} \cdot S_1 + c \cdot \sum_{j=0}^{i-1} 6^j = O\left((2i)^{\log_2(6)}\right),$$

where $c \leq 10$ is the constant overhead that arises from Equation (11). This concludes the proof of Claim 34. ◀

Proposition 33 now follows by combining Equation (10) and Claim 34. ◀