Electronic Colloquium on Computational Complexity, Report No. 81 (2023)

Constant-Round Arguments from One-Way Functions

Noga Amit*
Weizmann Institute of Science

Guy N. Rothblum†
Apple

Abstract

We study the following question: what cryptographic assumptions are needed for obtaining constant-round computationally-sound argument systems? We focus on argument systems with almost-linear verification time for subclasses of P, such as depth-bounded computations. Kilian's celebrated work [STOC 1992] provides such 4-message arguments for P (actually, for NP) using collision-resistant hash functions. We show that *one-way functions* suffice for obtaining constant-round arguments of almost-linear verification time for languages in P that have log-space uniform circuits of linear depth and polynomial size. More generally, the complexity of the verifier scales with the circuit depth. Furthermore, our argument systems (like Kilian's) are doubly-efficient; that is, the honest prover strategy can be implemented in polynomial-time. Unconditionally sound interactive proofs for this class of computations do not rely on any cryptographic assumptions, but they require a linear number of rounds [Goldwasser, Kalai and Rothblum, STOC 2008]. Constant-round interactive proof systems of linear verification complexity are not known even for NC (indeed, even for AC^1).

*Email: noga.amit@weizmann.ac.il. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702) and from the Simons Collaboration on The Theory of Algorithmic Fairness.

†Email: rothblum@alum.mit.edu. Part of this work was done while the author was at the Weizmann Institute of Science. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702) and from the Simons Collaboration on The Theory of Algorithmic Fairness.

ISSN 1433-8092

Contents

1 Introduction

A proof-system allows an untrusted prover to convince a verifier that a complex claim is true. The claim is usually framed as the membership of an input x in a language $\mathcal{L}$, where verification should be more efficient than deciding membership in $\mathcal{L}$. In a computationally sound *argument system* [BCC88], soundness is relaxed to hold only against polynomial-time cheating provers, under cryptographic assumptions. If $x \notin \mathcal{L}$, then no polynomial-time cheating prover should be able to get the verifier to accept (except with small probability). Understanding the cryptographic assumptions needed to construct argument systems, and the expressive power of these protocols (i.e., which languages have argument systems) is a central question in the foundations of cryptography.

We study this question, focusing on efficient argument systems that have a constant number of rounds and almost-linear communication and verification time, and on constructing such argument systems for subclasses of P, such as depth-bounded computations. Kilian's [Kil92] celebrated work showed that, assuming the existence of collision-resistant hash functions (CRHs), every language in P has a 4-message argument system with sublinear communication and almost-linear verification time (actually, this result applies to all of NP, but for now we focus on P and its subclasses). Kilian's result demonstrated that CRHs are sufficient for constructing argument systems that go well beyond what is known for *unconditionally sound* interactive proof systems (IPs) [GMR89] (and, in some regimes, beyond what is plausible for IPs [GH98, GVW02]). It is not well understood, however, whether such argument systems can be based on assumptions that are significantly weaker than collision-resistant hashing. The quest to understand the minimal assumptions needed for implementing cryptographic primitives is a central theme in the theoretical study of cryptography. Considering argument systems through this lens, we ask:

Can constant-round computationally sound argument systems with almost-linear communication and verification time be based on the "minimal"[1] assumption of one-way functions (OWFs)? Does their power extend beyond what is known (or plausible) using unconditionally sound IPs?

We answer these questions in the affirmative. Our main result is a constant-round argument system, whose security only relies on the existence of one-way functions, where the communication and the verification time grow linearly with the depth of the circuits computing the language:

Theorem 1.1 (Constant-round arguments from one-way functions)**.** *If one-way functions exist, then for every language $\mathcal{L}$ that is computable by log-space uniform circuits of fan-in 2, depth $D(n)$ and polynomial size, and for every desired constant $\sigma \in (0,1]$, there is a constant-round public-coin argument system, with perfect completeness and negligible soundness error against $\mathsf{poly}(n)$-time cheating provers, where n is the input length. The protocol's complexities are:*

- *Constant round complexity $O(1/\sigma^3)$,*[2]
- *Communication complexity $O(n^\sigma \cdot D(n))$,*
- *The verifier runs in time $O(n^\sigma \cdot D(n) + n^{1+\sigma})$, while the honest prover runs in $\mathsf{poly}(n)$ time.*

[1] One-way functions are often referred to as a "minimal" assumption for cryptography, and in many cases one-way functions are *essential* for constructing cryptographic primitives [IL89]. We note, however, that it is *not* known whether one-way functions are *essential* for constructing arguments. Wee [Wee05] studies the relationship between non-trivial arguments and various assumptions.

[2] By $O(1/\sigma^3)$ we mean that there exists a universal constant c s.t. the round complexity is at most c/σ^3.

The number of rounds is a constant, where this constant depends on the desired communication complexity and verification time. For linear-depth computations, for any desired constant σ, the communication and the verification time can be $O(n^{1+\sigma})$ using $O(1/\sigma^3)$ rounds. The protocol is *doubly-efficient*: the honest prover runs in polynomial time. Thus, this argument system can be used for delegating computation to an untrusted server, and obtaining a proof that a claimed output of the computation is correct [GKR15]. For simplicity, we take the security parameter to be a small polynomial in the input length throughout (obtaining security against $\mathsf{poly}(n)$-time adversaries). More generally, the communication, verifier runtime and prover runtime depend polynomially on the security parameter. Finally, having established that one-way functions suffice for extending the power of argument systems beyond what is known for interactive proof systems, a natural question for future work is whether one-way functions suffice for constructing arguments that can be verified in almost-linear time for all of P (or even for all of NP).

Comparison to known argument systems. As noted above, assuming the existence of CRHs, there exist 4-message doubly-efficient arguments with sublinear communication and almost-linear verification time for all of P [Kil92]. In fact, this celebrated protocol can be used for all of NP, i.e., a much richer class of computations. Indeed, such arguments exist even beyond NP, as was shown by Micali [Mic94] (under stronger assumptions) and by Barak and Goldreich [BG08]. One major focus, starting with [Mic94], is reducing the round complexity to "non-interactive" or to 2-message protocols, under minimal cryptographic assumptions. This vast literature is too vast for us to survey here. See e.g. Kalai, Raz and Rothblum [KRR22] and Choudhuri, Jain and Jin [CJJ21], as well as the recent expositions by Thaler [Tha22] and Ishai [Ish20a, Ish20b], and the references therein. Many of these works construct protocols for P or for subclasses of P. The vast majority of works on argument systems use assumptions that (at the very least) imply CRHs. One exception is works by Bitansky, Kalai and Paneth [BKP18] and by Komargodski, Naor and Yogev [KNY18], who construct argument systems based on the existence of the more relaxed primitive of multi-collision-resistant hash functions, though the recent work of Rothblum and Vasudevan [RV22] indicates that the gap between collision-resistance and multi-collision-resistance might not be wide.

The main distinction in our work is that *we rely only on the existence of one-way functions*, but our result is for a more restricted class of depth-bounded computations, and the round complexity, while constant, is larger than in Kilian's 4-message protocol (let alone the subsequent works that further reduce the interaction using stronger assumptions). OWFs are generally considered to be a considerably more relaxed assumption than CRHs. Simon [Sim98] showed a black-box separation between the two notions. We remark that while the long-standing open question of constructing CRHs from OWFs is well beyond the current state of the art, Holmgren and Lombardi [HL18] do show that an exponentially hard variant of OWFs is sufficient for constructing CRHs.

Comparison to Interactive Proofs. A parallel body of work studies the expressive power of unconditionally sound interactive proof systems. For the class of log-space uniform poly-size depth D circuits, known doubly-efficient IPs (DEIPs) require round complexity that is quasi-linear in D [GKR15], compared with the constant round complexity in our new protocol (though we note that our protocol is computationally sound, assumes the existence of one-way functions, and its communication complexity and verification time are larger by a small polynomial factor). See 1e 1 for a full comparison. The gap in the round complexity is not merely for known protocols: we find it quite plausible to conjecture that there do not exist constant-round interactive proofs for linear

depth computations (indeed, one could conjecture that they do not even exist for AC^1 circuits):

Remark 1.2 (IPs for linear depth.)**.** *In an interactive proof, we require a constant gap between the completeness and the soundness error. A significantly more relaxed requirement is to only have some infinitesimal gap between the completeness and the soundness error. Essentially all known results for IPs become straightforward if one is willing to make this relaxation (in particular, the gap can be smaller than the inverse of the computation size). Nonetheless, for linear-depth computations, it is not known how to construct a constant-round DEIP with* **any** *gap between the completeness and the soundness error. We find it plausible to conjecture that no such proof system exists. Indeed, even for* AC^1 *no such proof system is known. Of course, this assumption precludes the possibility of getting a DEIP with a* constant gap*, but our main result shows this* is *possible for an argument system (assuming OWFs). See Appendix A.1 for further discussion.*

In Table 1 we compare the power and complexity of known doubly-efficient IPs:

class (in)	# rounds	communication	verifier time	uniformity
depth D, size S [GKR15]	$O(D \cdot \log S)$	$D \cdot \mathsf{polylog}(S)$	$n \cdot \mathsf{poly}(D, \log S)$	log-space uniform
Space S, poly time [RRR16]	$\mathsf{exp}(\tilde{O}(1/\sigma))$	$n^\sigma \cdot \mathsf{poly}(S)$	$n^\sigma \cdot \mathsf{poly}(S) + \tilde{O}(n)$	none (Turing machine)
$\mathsf{AC}^0[\oplus]$ [GR20]	$O(1/\sigma)$	$n^{\sigma+o(1)}$	$n^{1+o(1)}$	adjacency predicate[3]
NC^1 [GR20]	$O(1/\sigma^2)$	$n^{\sigma+o(1)}$	$n^{1+o(1)}$	incidence function[4]
depth D, poly-size (this)	$O(1/\sigma^3)$	$O(n^\sigma \cdot D)$	$O(n^\sigma \cdot D + n^{1+\sigma})$	log-space uniform

Table 1: Comparison of doubly-efficient proof systems, where $\sigma > 0$ is a desired constant for bounding the communication. The proof systems from prior works are unconditionally sound, whereas our new result is computationally sound assuming the existence of one-way functions.

Elaborating on the relationship to known constant-round DEIPs: Reingold, Rothblum and Rothblum [RRR16] showed constant-round interactive proofs for languages that can be decided in bounded-polynomial space and polynomial time. This class is incomparable to the class of languages in our main result (languages with linear-depth polynomial-size circuits). Though the classes are incomparable, we view the round complexity in our construction as much smaller. Fixing a constant σ, to get communication complexity $O(n^\sigma \cdot \mathsf{poly}(S))$ for an S-space computation, the RRR protocol uses $\mathsf{exp}(\tilde{O}(1/\sigma))$ many rounds. In our protocol, on the other hand, $O(1/\sigma^3)$ rounds suffice for obtaining $O(n^\sigma \cdot D)$ communication for a depth-D computation. Goldreich and Rothblum [GR20] constructed constant-round protocols for highly uniform variants of the complexity classes

[3]The circuit's *adjacency predicate* should be computable by a $n^{o(1)}$-size formula that can be constructed in $n^{1+o(1)}$-time. See Section 3.3.

[4]The circuit's *incidence function* should be computable by a $n^{o(1)}$-size formula that can be constructed in $n^{1+o(1)}$-time. See Section 3.3.

$\mathsf{AC}^0[\oplus]$, using $O(1/\sigma)$ rounds, and NC^1, using $O(1/\sigma^2)$ rounds. The main distinction with our work is that our new protocol applies to computations well beyond NC^1. We remark, however, that the constant-round GR protocol for $\mathsf{AC}^0[\oplus]$ plays an important role in our construction.

1.1 Zero-Knowledge Arguments

Based on Theorem 1.1, we show that the existence of one-way functions suffices for succinct *constant round* zero-knowledge argument systems, with perfect completeness and constant soundness error, for NP relations whose verification circuit has bounded-polynomial depth. Succinctness means that the communication is nearly-linear in the witness length.

Theorem 1.3 (Constant-round succinct zero-knowledge for bounded-polynomial depth relations from one-way functions)**.** *If one-way functions exist, then for every language $\mathcal{L} \in \mathsf{NP}$ whose relation is computable by log-space uniform circuits of fan-in 2, depth $D(n)$ and polynomial size, and for every desired constant $\sigma \in (0, 1]$, there is a constant-round computational zero-knowledge argument system as follows. The argument system is public-coin and has perfect completeness and constant soundness error against $\mathsf{poly}(n)$-time cheating provers, where n is the input length. The protocol has constant round complexity $O(1/\sigma^3)$. Taking $M(n)$ to be the witness length, the communication complexity is $O(n^\sigma \cdot (M(n) + D(n)))$ and the verifier runtime is $O(n^\sigma \cdot (n + M(n) + D(n))$. Given a witness w for x's membership in $\mathcal{L}$, the honest prover runs in $\mathsf{poly}(n)$ time.*

Proof sketch. The zero-knowledge argument system follows from the public-coin protocol of Theorem 1.1, using a standard transformation for public-coins protocols [IY87, BGG+88]. The prover and the verifier run the protocol of Theorem 1.1 on the circuit computing the NP relation, with respect to input (x, w) (where x is the input and w is the witness). Rather than sending its messages in the clear, the prover sends computationally hiding and statistically binding bit commitments to the witness and to its messages. Such bit commitments can be constructed from one-way functions [Nao91]. After completing the protocol, the prover and the verifier run a non-succinct zero-knowledge proof to show that the verifier would accept if it saw the witness and the messages inside the prover's commitments (we need to use refinements to the 3-coloring protocol of [GMW91] to get constant soundness with communication and verification time that are nearly-linear in the circuit *size* for this final statement). □

The constant soundness error in Theorem 1.3 can be amplified, but we need to use sequential repetition to maintain zero-knowledge. Indeed, constructing even non-succinct constant-round zero-knowledge arguments with negligible soundness error from one-way functions is a long-standing open problem. We also remark that we could use *statistically hiding* and computationally binding commitments, to obtain an analogous succinct statistical zero-knowledge argument, but known constructions of such commitments from OWFs [HNO+09] are not constant-round (indeed, there are black-box lower bounds [HHRS15]). The resulting protocol would have a small polynomial number of rounds.

Comparison to prior work on succinct ZK from one-way functions. The comparison to the state of the art from prior work on constructing zero-knowledge from one-way functions is analogous to the comparison of Theorem 1.1 to prior work on DEIPs (all ZK protocols we compare to here are unconditionally sound proof systems). For NP relations computable by poly-size linear-depth circuits, the GKR protocol gives a succinct proof system with $\tilde{O}(n)$ rounds. The

RRR protocol gives constant-round succinct zero-knowledge for an incomparable class of poly-time bounded-polynomial space relations. The GR protocol imply constant-round succinct zero-knowledge for $\mathsf{AC}^0[\oplus]$ and NC^1 relations.

2 Technical Overview

We outline the key ideas underlying our constant-round argument construction (Theorem 1.1). We begin with a brief review on universal one-way hash functions (UOWHFs) and hash trees.

2.1 Background: UOWHFs, Local Openings and Holographic Proof-Systems

UOWHFs. A family $\mathcal{H}$ of universal one-way hash functions (UOWHFs), introduced by Naor and Yung [NY89], is a family of shrinking functions with the following property: fixing an input x, and drawing a random hash function h from the family $\mathcal{H}$, it is hard to find a "second preimage" $x' \neq x$ s.t. $h(x') = h(x)$. This is sometimes referred to as second-preimage collision resistance, or targeted collision resistance. Note that the order of events is important: the input x should be fixed *before* the hash function $h \sim \mathcal{H}$ is selected. This is a considerable relaxation to collision-resistant families, where even after h is chosen, it should be hard to find *any* collision (in a UOWHF, after h is revealed, it may well be possible to adaptively compute a pair x', x'' that collide, but they will not collide with any input that was fixed before h was chosen). Indeed, Rompel [Rom90] showed that UOWHFs can be constructed from any one-way function (Naor and Yung showed a construction from one-way permutations). Our construction uses a family of UOWHFs that map inputs in $\{0,1\}^{\kappa^2}$ to outputs in $\{0,1\}^{\kappa}$, where the security parameter κ is generally taken to be n^{ε} for a small constant $\varepsilon \in (0,1]$, and the "shrinkage" factor is $1/\kappa = 1/n^{\varepsilon}$.

Local opening. UOWHFs can be used in a hash tree to hash an M-bit string $x \in \{0,1\}^M$ to a short commitment (or hash root) $y \in \{0,1\}^{\kappa}$. The root y can later be used to *locally open* any desired bit x_i. The construction divides the string x into "chunks" of length κ and places them on the leaves of a binary tree of depth $\ell \approx \log(M/\kappa)$ (i.e., at layer ℓ of the tree). For each internal node in the tree, its value is the output of a UOWHF applied to (the concatenation of) its children's values. Thus, nodes are hashed up the tree, and the value at the root is the commitment or hash-root. Later, we can "open" $x[i]$ (the i^{th} bit of x) by revealing the hash values along the path from the root to the leaf containing the i^{th} bit, together with the value of the sibling of each node along this path. The two important properties of this construction are:

1. **Local opening:** given any i, the local opening for $x[i]$ only requires sending $O(\kappa \cdot \log M)$ bits and can be verified in $\mathsf{poly}(\log M, \kappa)$ time.

2. **Local targeted collision resistance:** for any string x fixed before choosing the hash functions (see below), taking $\mathsf{root}(x)$ to be the (correct) hash root according to x, it is hard to find an index i and a valid opening for any value of the i^{th} bit that is different from $x[i]$.

Bellare and Rogaway [BR97] observed that using a single UOWHF in a straightforward hash tree [Mer89, Dam89] might not be secure. However, the construction is secure if we use a separate hash function for each layer of the tree ($(\ell - 1)$ UOWHFs in all).

Our construction uses a d-ary hash tree, which is a straightforward extension of the binary tree described above. The depth is $\ell \approx \log_d(|x|/\kappa)$, and local opening requires sending $O(\ell \cdot d \cdot \kappa)$ bits.

We comment that this construction is not a "standard" commitment scheme in the sense that it is not necessarily hiding (and, as described in the second item, only satisfies the relaxed "targeted" binding property when comparing to the one implied by CRH).

Holographic proof systems. In a *holographic* proof system, the verifier is not given access to the main input explicitly. Instead, it outputs a claim about the *encoding of the input* under a high distance error correcting code. Given this redundancy, the verifier should run in sublinear time in the input length. Holographic proof systems were introduced by Babai *et al.* [BFLS91] in the context of PCPs. Holographic Interactive Proofs were formalized and generalized by Gur and Rothblum [GR17].

In our work, the code used for the "holographic input" x will always be the low-degree extension (LDE, see Section 3.2). After interacting with the prover, the verifier either rejects, or it outputs a claim (r, v) about the input's encoding $\mathsf{LDE}(x)$, where r is a location in $\mathsf{LDE}(x)$, and v is a claim about the value of $\mathsf{LDE}(x)$ at location r. Completeness means that if the prover's statement is true and the prover follows the protocol, then the verifier doesn't reject and it holds that $\mathsf{LDE}(x)[r] = v$. Soundness means that if the prover's statement is false, then the probability that the verifier doesn't reject and $\mathsf{LDE}(x)[r] = v$ is small. The statement can either be that the input x is in a language $\mathcal{L}$, or that $f(x) = y$ for a specified function f and claimed output y (verifying membership in a language corresponds to the case where f is a Boolean-valued function).

Holographic interactive proofs are at the heart of many IP systems. In particular, all the proof systems in Table 1 have holographic variants, where the verifier's runtime is reduced to being nearly-linear in the communication complexity, while the number of rounds, communication complexity, and prover runtime are unchanged.[5] If the claim is about evaluating a function f with output length $|y|$ (rather than membership in a language), then the communication complexity and verification time grow by an additive term that is nearly-linear in $|y|$.

2.2 An Argument-System Template

We begin by describing a "template protocol" for constructing argument systems, which allows us both to prove a warm-up for our main result, and also to introduce important ideas and concepts from the full protocol. In particular, we isolate a new primitive: a *holographic hash root* protocol, which suffices for constructing argument systems that have a small number of interaction rounds.

The warm-up in this section gives an argument system for linear-depth circuits with round complexity $\mathsf{exp}(\tilde{O}(1/\sigma))$, and nearly-linear communication and verification time. We note that while the number of rounds is an exponentially larger constant than in our main result, it already goes well beyond what is known (and, under plausible assumptions, beyond what is possible) using unconditionally sound interactive proofs. Let C be a log-space uniform circuit ensemble with size $S = S(n) = \mathsf{poly}(n)$ and depth $D = D(n)$. Without loss of generality, we assume that the circuit is layered, where the gates in layer i are the ones at distance $(i-1)$ from the circuit's output gate, and for each i, the gates in layer i are fed (only) by gates in layer $(i+1)$. We also pad the circuit so that each layer is exactly of width S. On input x, the template protocol proceeds as follows:

1. The verifier chooses UOWHFs $\bar{h}$ for a hash tree on $M = \mathsf{poly}(S)$ bits and sends them to the prover.

[5] The GR proof-system for AC^0 is not holographic as-is, but modifying it to be holographic is straightforward, see Appendix A.2.

2. For each layer i of the circuit C, let $V_i \in \{0,1\}^S$ be the values of the gates in layer i when the circuit is evaluated on the input x. Let $\widehat{V}_i \in \{0,1\}^M$ be the encoding of the i^{th} layer using the low-degree extension (see above). For each $i \in [1, \ldots, D-1]$, the prover computes $\widehat{V}_i$, hashes it using the hash tree, and sends the root $y_i = \mathsf{root}(\widehat{V}_i)$ to the verifier.

3. The verifier receives alleged hash roots $\{\widetilde{y}_i\}_{i=1}^{D-1}$. The prover and the verifier run in parallel $(D-1)$ executions of an unconditionally sound holographic interactive proof (HIP, see above). The i^{th} execution is on (holographic) input V_{i+1} (the values of gates at layer $(i+1)$), and proves that $\widetilde{y}_i$ is the correct hash root for the low-degree extension of the values of the gates *in the i^{th} layer*, where these latter values (of layer i) are computed by applying the gates in the i^{th} circuit layer to the string V_{i+1} that is the input to the proof system.

   The outputs of these $(D-1)$ executions are claims $\{(r_i, v_i)\}_{i=2}^{D}$, where the i^{th} claim alleges that the value of the low-degree extension $\widehat{V}_i$ at location r_i has value v_i. We also add the claim that the circuit accepts as a claim (r_1, v_1) about the LDE of the output layer.

4. The verifier accepts if the following checks pass:

   (a) none of the HIP executions rejected.

   (b) the verifier asks the prover to perform a local opening for each $\widetilde{y}_i$: opening the r_i^{th} location in the hashed string, and showing that its value is v_i.

   (c) For the input layer, the verifier also checks that $\widehat{x}[r_D] = v_D$ (this requires evaluating the input's low-degree extension at a single point).

Completeness and Soundness. Completeness follows by construction. For soundness, let $\widehat{V}_i$ be the correct LDE of the gates in layer i of the circuit (when evaluated on the input x), and consider the hash roots $\{\widetilde{y}_i\}_{i=1}^{D-1}$. A critical insight in our analysis is that the values $\{\widehat{V}_i\}$ *are fixed before the verifier chooses its hash functions.* Thus, the hash tree's local targeted collision resistance applies, and for each i, if $\widetilde{y}_i$ is "correct", i.e. if it is the real value at the root of the hash tree on $\widehat{V}_i$, then in Step (4b), the prover cannot open $\widetilde{y}_i$ to any other value except $\widehat{V}_i[r_i]$. This will be quite helpful for catching the prover if it is cheating.

We proceed as follows: if the first hash root $\widetilde{y_1}$ is correct, then the prover has committed to a string indicating that the circuit rejects the input! This commitment is binding, and in particular the verifier will reject when it checks the opening of the commitment in Step (4b). Otherwise, if $\widetilde{y_1}$ is incorrect, then there are two possibilities: either there is some layer $i^* \in [1, \ldots, D-2]$ where $\widetilde{y_{i^*}}$ is incorrect, but $\widetilde{y_{i^*+1}}$ is correct. The soundness of the HIP implies that the $i^{*\text{th}}$ execution will yield a false claim, i.e. $\widehat{V_{i^*+1}}[r_{i^*+1}] \neq v_{i^*+1}$. But since the prover sent the correct hash root for the $(i^*+1)^{\text{th}}$ layer, in Step (4b), it cannot open the hash root to any value except the correct value (which is different from v_{i^*+1}), and the verifier will reject. The remaining possibility is that the prover was cheating on $\widetilde{y_{D-1}}$: in this case, w.h.p. the HIP for layer $(D-1)$ outputs a false claim about the LDE of the input x, and the verifier will reject in Step (4c).

Complexity analysis. The communication complexity is D times the communication complexity in the HIP, plus $(D \cdot \mathsf{poly}(\log(n), \kappa))$ for sending the hash roots and openings. Similarly, the verifier runtime is D times the runtime in the HIP, plus $(D \cdot \mathsf{poly}(\log(n), \kappa))$ for the hash roots, and openings and another almost-linear term for the final LDE check in Step (4c) (this final term can be omitted

if we give the verifier query access to the LDE of the input x). Finally, the round complexity is dominated by the round complexity of the HIP executions of Step (3), but the key point is that these are *performed in parallel*, and the proof in each execution is for a computation whose depth is independent of the depth of the circuit C.

The HIP. The i^{th} HIP is performed to check the following computational claim: let C_i^{layer} be the circuit that computes the i^{th} layer of the circuit C. Let LDE be the circuit that takes an input $V \in \{0,1\}^S$ and computes its low-degree extension $\widehat{V} \in \{0,1\}^M$. Finally, let root be the circuit that takes an M-bit string $\widehat{V}$ and outputs the root of the hash tree computed on $\widehat{V}$ (with the hash functions $\bar{h}$ sent by the verifier). The i^{th} execution uses the HIP to verify a claim about the value of the function $(\mathsf{root} \circ \mathsf{LDE} \circ C_i^{\mathsf{layer}})$, where we think of the input to this function as the values of the $i+1^{\text{st}}$ layer of the circuit. Any HIP that can perform this computation efficiently may be used here. For example, we can take the holographic variants of the GKR or RRR protocols (see Table 1, and note that the verifier time is reduced in the holographic case). In particular, since this function can be computed in polynomial time and $\mathsf{poly}(\kappa)$ space, RRR can yield $\mathsf{exp}(\tilde{O}(1/\sigma))$ rounds, $(n^{\sigma+o(1)} \cdot \mathsf{poly}(\kappa))$ communication and verification time, and polynomial prover time. Alternatively, since this function is also computable by log-space uniform circuits of depth $\mathsf{poly}(\kappa)$, GKR can yield $\mathsf{poly}(\log n, \kappa)$ rounds, communication, and verification time. Under stronger cryptographic assumptions, one could also use the GR protocols, see Remark 2.1.

Digest. Our protocol uses the UOWHF to force the prover to send a commitment for each layer of the circuit, where a cheating prover has two (bad) choices, either (a) send a correct hash root for that layer's gate values. In this case, the commitment is binding, and the prover's hands are forevermore tied when it makes claims about this layer's LDE, or (b) the prover sends an *incorrect* hash root, where at the very least the prover needs to send an incorrect hash root for the output layer (otherwise it will be caught immediately). Since there must be *some* layer where the prover is cheating on the root of layer i but we can access the correct LDE of layer $(i+1)$ (either because the prover sent a correct hash root, which is binding, or because layer $(i+1)$ is the input layer), verification can be reduced to checking consistency between a hash root and the layer below it. I.e., we have reduced verifying the deep / complex computation of C, to verifying (in parallel) many simpler computations. Each of these simpler computations evaluates one circuit layer, and composes it with a computation of the low-degree extension and the hash tree. Thus, our goal is constructing efficient HIPs for these simpler computations (moreover, as we will see below, these simpler computations have nice structure that facilitates the construction of very efficient proof systems).

2.3 Holographic Hash Root (HHR) Protocol

In a *holographic hash root (*HHR*)* protocol, the prover and the verifier are given a claim of the form $(\bar{h}, y)$ and a holographic input w. After interacting with the prover, the verifier (who never accesses w) either rejects or outputs a claim (r, v) about the LDE $\widehat{w}$ of w. If y is the correct hash-root of w w.r.t. the hash functions $\bar{h}$, then $\widehat{w}[r] = v$. If y is *not* the correct hash root, then w.h.p. either the verifier rejects, or $\widehat{w}[r] \neq v$. The HHR protocols in this work have information-theoretic soundness (though computational soundness would suffice for the template protocol).

On a conceptual level, our work identifies HHR protocols as a very useful component for constructing argument-systems with small round-complexity. Once we have a HHR protocol, we can

compose it sequentially with a HIP for verifying a claim about the computation that takes as input a vector V of values for the gates at layer $(i+1)$, computes the values V' that V induces for the gates in layer i, and checks a single claim about the LDE of V'. We can construct a constant-round HIP for the latter task (evaluating a single circuit layer and then computing a low-degree extension) using the GR protocol (see Table 1). Thus, *the round complexity of the template protocol is dominated by the round complexity that can be achieved for* HHR*s.*

A better HHR protocol. Our main technical contribution is an HHR protocol whose round complexity is only $(1/\sigma^3)$. Our main result (Theorem 1.1) follows by plugging the HHR protocol into the template protocol of Section 2.2. The HHR protocol closely follows the construction of a hash tree. Fixing a small constant $\delta > 0$ (set below), we use a family of UOWHF functions $\{h : \{0,1\}^{n^{2\delta}} \to \{0,1\}^{n^{\delta}}\}_{h \in \mathcal{H}}$ (the security parameter κ is set to be a sufficiently small power of n). We use these hash functions in an $d = n^{\delta}$-ary hash tree (see Section 2.1). The tree has $\ell = O(1/\delta)$ layers, where layer j in the tree is n^{δ}-times smaller than layer $j+1$. Given the root of the tree, local opening requires sending $O(n^{2\delta})$ bits to the receiver/verifier (for each node on the path from the root to the leaf, the values of all $(d-1)$ of its siblings need to be sent).

The HHR protocol sequentially "strips away" the layers of the hash tree, beginning with a claim about the hash root and ending with a claim about the leaves of the tree. This is achieved by means of a *tree-layer sub-protocol*: a $O(1/\delta^2)$-round protocol that begins with an input claim about the LDE of the tree nodes in layer j, and ends with an output claim about the LDE of the tree nodes in layer $(j+1)$. If the input claim is correct and the prover follows the protocol, then the output claim will also be correct. If, however, the input claim is incorrect, then (no matter what strategy a cheating prover utilizes) the output claim will also be incorrect. This structure is inspired by, and similar to, the GKR protocol, but we emphasize that the computation for moving from one layer to another is *not* of constant depth, since it involves applying the hash function, which is an arbitrary $\mathsf{poly}(\kappa) = n^{O(\delta)}$-time computation, whereas we want a $O(1/\delta^2)$ round protocol.

Several remarks are in order. We emphasize that we run the tree-layer sub-protocols *sequentially*, starting from the output layer (layer 1), and ending with the bottom of the tree (layer $(\ell-1)$). There are $O(1/\delta)$ tree layers, so the total round complexity is $O(1/\delta^3)$. Theorem 1.1 is derived by taking δ to be a small enough constant multiple of the desired σ, so the $n^{O(\delta)}$ term in the verification time and communication complexity (see Theorem 5.1) ends up being n^{σ}. Finally, the alert reader will have noticed that we need to begin the HHR with a claim about the LDE of the hash root, and we will end it with a claim about the LDE of the values in the leaves. For the first point: the verifier, who knows the claimed hash root y, can choose a random location r and take $v = \mathsf{LDE}(y)[r]$ to be the input claim to the first sub-protocol. If y is not the correct hash root, then w.h.p. over the choice of r the first input claim will be false (since the low-degree extension is a high-distance error-correcting code). Second, by definition of the HHR, the values in layer ℓ are already a low-degree extension of the string w. In the final sub-protocol, we will directly get a claim about $\mathsf{LDE}(w)$ (rather than a claim about $\mathsf{LDE}(\mathsf{LDE}(w))$.

Remark 2.1 (Stronger assumptions)**.** *If the UOWHF were computable in highly uniform* $\mathsf{AC}^0[\oplus]$*, we could instead simply use the constant-round GR protocol to move from one layer to another (see Table 1). In fact, a UOWHF computable in highly uniform* NC^1 *should suffice, since the GR protocol can also work on highly uniform* NC^1 *circuits. Indeed, it is possible to prove that the circuit that computes the UOWHF tree can satisfy this stronger uniformity condition (see Footnotes 3 and*

4 for the exact uniformity conditions), and it would be an NC^1 *circuit thanks to the polynomial shrinkage of the UOWHF, that promises that the tree has a constant number of layers.*

UOWHFs in such low classes have been conjectured to exist (see e.g. [AM13], but note that we need super-linear shrinkage in our construction, because we want the tree to be of constant depth). Regardless, in this work, we do not want to assume anything beyond the existence of one-way functions.

2.3.1 The Tree-Layer Sub-Protocol

We briefly sketch some of the ideas in this final sub-protocol. Let $n_{in} = n^{2\delta}$ and $n_{out} = n^{\delta}$ be the input and output lengths of the hash function. We take $w_i \in \{0,1\}^{n^{i\cdot\delta}}$ to be the vector of the values of nodes in layer i of the hash tree, and let $k = k(i) = |w_{i+1}|/n_{in}$ be the number of nodes in layer i. As described above, given a claim (i, r_i, v_i) about w_i, and given also a holographic input w_{i+1}, the goal of our sub-protocol is for the verifier (to reject or) to output a claim (r_{i+1}, v_{i+1}) about the holographic input. The crux of the matter is doing this using only $O(1/\delta^2)$ rounds, which we accomplish by utilizing the particular structure of the hash tree's computation: the tree operates independently and in parallel on blocks of w_{i+1}. Dividing the layers into blocks we have:

$$w_i = y_1, ..., y_k \text{ for } |y_j| = n_{out},$$

where y_j is the value of the j^{th} node in layer i, and

$$w_{i+1} = z_1, ..., z_k \text{ for } |z_j| = n_{in},$$

where each z_j is the concatenation of the values of the j^{th} node's children. Taking h_j to be the hash function for layer j, we can now restate the claim about layer j:

$$\left(\bigwedge_{j=1}^{k} y_j = h_j(z_j)\right) \bigwedge (\mathsf{LDE}(y_1, \ldots, y_k)[r_i] = v_i)\,. \tag{1}$$

Thus, there are k "mini-claims", each about a single evaluation of the hash function, tied together by a " global claim" about the low-degree extension of (the concatenation of) the resulting outputs. We use a *batch-verification* protocol to verify the k mini-claims, together with the global claim about the LDE, at a cost that is not much larger than verifying a single claim (each single claim is about single a $\mathsf{poly}(\kappa)$-size computation, so the verifier can verify it on its own). Our protocol is inspired by the UP batching protocol of Reingold, Rothblum and Rothblum [RRR18]. The idea is to proceed in sequential iterations, where in each iteration we run a "reducing" sub-sub-protocol to restrict the claims being made to a smaller subset $S' \subseteq [k]$ of the k initial mini-claims, tied together with a "global claim" about the computations in the set S'. The size of the set is reduced by a factor of roughly n^{δ} in each iteration, so after $O(1/\delta)$ iterations, the final set has only a few surviving mini-claims. The prover can send to the verifier the values of the surviving tree nodes, and the verify can verify the remaining claims by brute force in $n^{O(\delta)}$ time and communication (there is a technical issue here: this is a holographic protocol, so we need to reduce these final mini-claims to claims about the LDE of the $(i+1)^{\text{st}}$ tree layer).

As in [RRR18], the "reducing" sub-sub-protocol is performed using an interactive proof of proximity (IPP) [RVW13], where a claim about a large implicit input X (the sequence of y_i's and z_i's) is reduced to a claim about a subset of X's bits. We elaborate briefly on how this is done in

our context. We use an IPP where the verifier (on top of having implicit input) has *holographic* input, and at the end of the interaction, the verifier outputs a claim about its encoding. We view the k hash outputs (in layer i) as the implicit input, and the k hash inputs (in layer $(i+1)$) as the holographic input. The IPP lets us reduce a claim about a set S of input-output pairs to: (i) a claim of the same form about a smaller subset of the pairs, and (ii) a holographic claim about the encoding of the inputs. We "set aside" the holographic claims generated by the IPPs, and at the end of the protocol we reduce all of them to a single claim about the LDE of the $(i+1)^{\text{st}}$ layer. The reducing sub-sub-protocol has $O(1/\delta)$ rounds, $n^{O(\delta)}$ communication and verification time, and a polynomial prover. Rolling these complexities back to the HHR protocol and the template protocol gives the result claimed in Theorem 1.1.

We remark that there are significant technical hurdles that need to be overcome in the full construction. The main reason is that we want the reducing sub-sub-protocol to run in only $O(1/\delta)$ rounds. Thus, we can only afford to use (an extension of) the GR protocol for highly-uniform $\mathsf{AC}^0[\oplus]$ circuits in the IPP.[6] Thus, we need to carefully argue that all the computations being verified can be performed via highly uniform low-complexity circuits. For example, we need to augment the implicit y_i inputs in the IPP with the entire tableau of the hash function's computation, so that verification can be in $\mathsf{AC}^0[\oplus]$. We also need to carefully argue about the structure of the "global claims" tying together the mini-claims in each iteration, to ensure they can be verified by a highly-uniform low-depth circuit.

This concludes our high-level sketch of the sub-sub-protocol's structure, and we direct the reader to Section 4 for a more detailed overview and the full details.

Organization. Preliminaries and technical definitions are in Section 3. In Section 4, we prove the HHR protocol. The full proof of Theorem 1.1, that is, a constant-round argument from one-way functions, is in Section 5. We refer to this proof system as "flat-GKR", as it essentially "flattens" the GKR protocol, and performs its consistency checks between circuit layers *in parallel*, instead of *sequentially*.

3 Preliminaries

For a string $x \in \Sigma^n$ and an index $i \in [n]$, we denote by $x[i]$ the i^{th} entry in x. For a set $I \subseteq [n]$, we use $x|_I$ to describe the sequence of entries in x corresponding to coordinates in I, and "$\circ$" for string concatenation. For a finite field $\mathbb{F}$ and a string $x \in \mathbb{F}^*$, we use $[x]_2$ for denoting x's binary representation. There are many possible representations, and we fix one of them (we formally define it where it is used). Notice that this representation is an injection from the field $\mathbb{F}$ to $\{0,1\}^{\lceil \log(|\mathbb{F}|) \rceil}$, and we do not argue that field operations are preserved under it.

For any pair of distributions, $D_1 \equiv D_2$ means that they are identical. We use $x \in_R D$ to indicate that the element x was drawn uniformly at random from the distribution D.

Let $x, y \in \Sigma^n$ be two strings of length $n \in \mathbb{N}$ over a (finite) alphabet Σ. We define the (relative Hamming) distance of x and y as $\Delta(x,y) \stackrel{\text{def}}{=} |\{x_i \neq y_i : i \in [n]\}|/n$. If $\Delta(x,y) \leq \varepsilon$, then we say that x is ε-close to y, and otherwise we say that x is ε-far from y. We define the distance of x from a (non-empty) set $S \subseteq \Sigma^n$ as $\Delta(x,S) \stackrel{\text{def}}{=} \min_{y \in S} \Delta(x,y)$. If $\Delta(x,S) \leq \varepsilon$, then we say that x is

[6] In fact, $\mathsf{AC}^0[\oplus]$ isn't a sufficiently rich for our purposes, so we extend the protocol to apply to constant-depth *arithmetic* circuits over large fields with bounded fan-in multiplication gates (see Appendix A.2).

ε-close to S and otherwise we say that x is ε-far from S. We extend these definitions from strings to functions by identifying a function with its truth table. For a set S (of size at least 2), take its minimum distance to be the minimum, over all distinct vectors $x, y \in S$ of $\Delta(x, y)$.

3.1 UOWHF and Merkle Tree

Definition 3.1 (UOWHF [NY89]). *Let κ be a security parameter, and let $\{n_{1_i}\}$ and $\{n_{0_i}\}$ be two polynomially related increasing sequences that depend on κ, such that for all $i, n_{0_i} \leq n_{1_i}$. Let $\mathcal{H}_k$ be a collection of functions such that for all $h \in \mathcal{H}_k, h : \{0,1\}^{n_{1_k}} \to \{0,1\}^{n_{0_k}}$.*

Let A be a probabilistic $\mathsf{poly}(\kappa)$-time algorithm that on input 1^k outputs $x^{(1)} \in \{0,1\}^{n_{1_k}}$ that we call an initial value*, then given a random $h \in \mathcal{H}_k$ attempts to find $x^{(2)} \in \{0,1\}^{n_{1_k}}$ such that $h(x^{(1)}) = h(x^{(2)})$ but $x^{(1)} \neq x^{(2)}$. Such an $\mathcal{H} = \mathcal{H}_k$ is called a* family of universal one-way hash functions (UOWHFs) *if for all $\mathsf{poly}(\kappa)$-time probabilistic algorithms A, the following holds for sufficiently large k:*

1. *If $x^{(1)} \in \{0,1\}^{n_{1_k}}$ is A's initial value, then $\Pr[A(h, x^{(1)}) = x^{(2)}, h(x^{(1)}) = h(x^{(2)}), x^{(2)} \neq x^{(1)}] = negl(n_{1_k})$ where the probability is taken over all $h \in \mathcal{H}$ and the random choices of A.*
2. *$\forall h \in \mathcal{H}$ there is a description of h of length polynomial in n_{1_k}, such that given h's description and x, $h(x)$ is computable in polynomial time.*
3. *$\mathcal{H}$ is accessible: there exists an efficient algorithm G such that on input 1^k, G generates uniformly at random a description of $h \in \mathcal{H}$.*

We note that we treat $\mathcal{H}$ as a collection of descriptions *of functions.*

We comment that the uniformity condition that A satisfies will eventually determine the uniformity condition that the prover in the argument system of Section 5 satisfies: If the UOWHFs are secure against non-uniform adversaries, then the protocol is secure against non-uniform provers, whereas if the UOWHFs are secure only against uniform adversaries, then the protocol is as well.

As mentioned in the Overview (see Section 2), Rompel gave the first construction of UOWHFs from arbitrary one-way functions, while Katz and Koo gave the first full proof for Rompel's construction.

Theorem 3.2 ([Rom90, KK05]). *The existence of one-way functions implies the existence of universal one-way hash functions.*

In what follows, we define a Merkle Tree and a valid path in the standard way. Next, we define a "relaxed" (when comparing to CRHs) security property that a commitment scheme based on a Merkle tree may satisfy, and we call such a commitment a "2-PLOSC" (Definition 3.5). We show in Claim 3.6 that instantiating a Merkle tree with a family of UOWHFs results in a 2-PLOSC. We conclude that the existence of one-way functions suffices for constructing a 2-PLOSC. As was mentioned in the overview, we stress that this construction is not a "standard" commitment scheme in the sense that it is not necessarily hiding, and that its security is a "targeted" binding property.

Construction 3.3 (Merkle Tree). *Fix $N, n_{in}, n_{out} \in \mathbb{N}$ and a finite alphabet Σ. Set[7] $L = \frac{N}{n_{in}}$, and let $\ell = \ell(N, n_{in}, n_{out})$ to be defined below. Given a string $z \in \Sigma^N$ and functions $h_1, ..., h_\ell : \Sigma^{n_{in}} \to \Sigma^{n_{out}}$, the* Merkle Tree *$T = T(z, h_1, ..., h_\ell)$ is defined in the following manner:*

[7] We assume that N, as well as any other layer length, is a multiple of n_{in}. This is always possible by padding, and does not hurt the security since the all-zero string is a fixed string, and therefore the UOWHF security condition applies to it as well.

- *The tree has $\ell+1$ layers. The first layer is the root and the last layer is the leaves;*
- *There are L leaves. For $j=1,...,L$, the j^{th} leaf is denoted by $c_{(\ell+1,j)}$ and contains $z[(j-1)n_{in}]...z[j\cdot n_{in}-1]$;*
- *For $i\in[\ell]$, the i^{th} layer is denoted by w_i and is created by applying h_i on the $i+1^{st}$ layer: The j^{th} block in the i^{th} layer is denoted by $c_{(i,j)}$ and contains $h_i(c_{(i+1,j)})$ for $j=1,...,\frac{|w_{i+1}|}{n_{in}}$;*
- *The root is denoted by $y=h_1(c_{(2,1)})$, thus $w_1=y$.*

We comment about the indexing of the tree nodes and blocks. As the tree is (n_{in}/n_{out})-ary, it is most convenient for us to index "chunks" of n_{in} nodes, which we call a *block*: namely, for each fixed tree layer i, the blocks in this layer are indexed as $c_{(i,1)},\ldots,c_{(i,M)}$ for $M=|w_i|/n_{in}$. In other words, each tree node is coupled together with its $(n_{in}/n_{out}-1)$ siblings. We stress that we do not give an explicit notation for each tree *node*, only to blocks of nodes. In particular, this means that there is no explicit indexing for the image of each block, e.g., for $h_i(c_{(i+1,j)})$, but we will not need one. In comparison to a binary tree, this indexing is equivalent to coupling each pair of nodes (that are siblings, i.e., mapped together to the next layer by the hash function) to a block, and index the block only.

Definition 3.4 (Valid Path)**.** *Let $N,n_{in},n_{out},\Sigma,L,\ell$ and $h_1,...,h_\ell$ be as in Construction 3.3. Given a string $y\in\Sigma^{n_{out}}$ and a leaf index $q'\in[L]$, a path $p=(p_1,...,p_{\ell+1})$ is called a* valid path from q' to y *if it satisfies the following properties:*

- $\forall i\in[\ell+1]:p_i\in\Sigma^{n_{in}}$;
- $\forall i\in[\ell]:p_i=h_i(p_{i+1})$;
- $p_1=y$.

Definition 3.5 (2-PLOSC)**.** *Let $N,n_{in},n_{out},\Sigma,\ell$ and $h_1,...,h_\ell$ be as in Construction 3.3. Assume that $n_{out}\ll N$. A* Second-Preimage Locally Openable Succinct[8] Commitment using a Merkle tree *for strings in Σ^N is defined via a pair of probabilistic polynomial-time algorithms $(\mathcal{S},\mathcal{R})$ such that:*

- **Commit Phase:**
  - *$\mathcal{S}$ chooses a string $z\in\Sigma^N$;*
  - *$\mathcal{R}$ sends hash functions $h_1,...,h_\ell:\Sigma^{n_{in}}\to\Sigma^{n_{out}}$;*
  - *$\mathcal{S}$ sends a commitment $y\in\Sigma^{n_{out}}$: the (correct) hash root of the Merkle Tree $T=T(z,h_1,...,h_\ell)$.*
- **Local-Opening Phase:** *$\mathcal{S}$ outputs an index $q\in[N]$ and an opening of a leaf $q'=\left\lceil\frac{q}{n_{in}}\right\rceil$, that is, a path $p=(p_1,...,p_{\ell+1})$.*
- **Security:** *For a sufficiently large n_{in} and for all $q'=\left\lceil\frac{q}{n_{in}}\right\rceil$,*

$$\Pr_{\substack{\mathcal{S}\ coins\\ h_1,...,h_\ell}}[p \text{ is a valid path from } q' \text{ to } y \text{ and } p_{\ell+1}\neq z[(q'-1)n_{in}]...z[q'\cdot n_{in}-1]]=negl(n_{in}).$$

[8] We call this scheme *succinct* because the output of the commit phase is small.

Claim 3.6 (A UOWHF tree is a 2-PLOSC)**.** *In the setting of Definition 3.5, if R samples $h_1, ..., h_\ell$ uniformly at random from a UOWHFs family*[9] *$\mathcal{H} : \Sigma^{n_{in}} \to \Sigma^{n_{out}}$, and n_{in} and N are polynomially related, then the commitment scheme is secure. In other words, given the functions, the leaves and the correct root for them, it is impossible for $\mathcal{S}$ to find an index and a second valid opening for it.*

Proof. Assume towards contradiction that there exist $N, n_{in}, n_{out} \in \mathbb{N}$, a finite alphabet Σ and a PPTM (probabilistic polynomial-time Turing machine) $\mathcal{S}$, such that $\mathcal{S}$'s success probability in breaking the security of the commitment is some non-negligible function $\eta(n_{in})$. As before, set $L = N/n_{in}$, and ℓ to be the number of layers. Denote the number of blocks in the tree by N. We use $\mathcal{S}$ to build a PPTM $\mathcal{A}$ that breaks the second-preimage resistance of one of the UOWHFs with some non-negligible probability. $\mathcal{A}$ simulates the receiver $\mathcal{R}$ in the commitment scheme, and works as follows:

- **$\mathcal{A}$ chooses the first preimage:**
  - It samples[10] a block index (i, j) out of all N blocks in the tree (its "guess" for the location of the collision);
  - It interacts with $\mathcal{S}$ and they perform the first step of the Commit Phase, in which $\mathcal{S}$ chooses a string $z \in \Sigma^N$;
  - It samples $\ell - 1$ function from $\mathcal{H} : h_1, ..., h_{i-2}, h_i, ..., h_\ell$;
  - Using $h_\ell, ..., h_i$ and z, it builds the Merkle Tree T up to its i^{th} layer;
  - $\mathcal{A}$ declares $x^{(1)} = c_{(i,j)}$ as its initial value.
- **$\mathcal{A}$ receives the function**: a function h is sampled from $\mathcal{H}$ and sent to $\mathcal{A}$.
- **$\mathcal{A}$ computes a second preimage:**
  - $\mathcal{A}$ resumes the interaction with $\mathcal{S}$. Following the second and third steps of the Commit Phase, $\mathcal{A}$ sends the hash functions $h_1, ..., h_{i-2}, h, h_i, ...h_\ell$ to $\mathcal{S}$, which sends back a commitment y;
  - They proceed to the Local-Opening Phase: $\mathcal{S}$ outputs an index $q \in [N]$ and an opening of a leaf $q' = \left\lceil \frac{q}{n_{in}} \right\rceil$, that is, a path $p = (p_1, ..., p_{\ell+1})$;
  - $\mathcal{A}$ declares $x^{(2)} = p_i$ as its second preimage.

Correctness. Let $\mathcal{A}$'s guess (i, j) and $z, h_1, ..., h_\ell, y, q', p$ be as described above. Notice that the maximal number of blocks in each layer is L, as the last layer is the longest. We define $\tau : [L] \times [\ell + 1] \to [L]$ in the following manner: Given a leaf index j and a layer i, consider the unique path from the root to the j^{th} leaf. $\tau(j, i)$ is the block index of the i^{th} layer that this path passes through.

We denote by p' the unique path in T from the root to the leaf in the index q'. It is composed out of tree blocks, where $c_{(i,j_i)}$ is the i^{th} block, namely:

$$p' = \left(c_{(1,j_1)}, ..., c_{(\ell+1,j_{\ell+1})}\right),$$

[9] Definition 3.1 considers a UOWHF as a Boolean function. Here we allow the functions to be over some finite alphabet Σ, and the definition is extended in the natural way. In our protocol, however, we take $\Sigma = \mathbb{GF}[2]$.

[10] We think of the sampling process as follows: $\mathcal{A}$ samples a block $v \in [N]$. It then finds the unique (i, j) to which v corresponds in the full binary tree, assuming some canonical indexing of the blocks.

where $\forall i \in [\ell+1], j_i = \tau(q', i)$. In particular, $j_{\ell+1} = q'$ and $j_1 = 1$, hence $p' = \left(c_{(1,1)}, ..., c_{(\ell+1,q')}\right)$.

Our aim is to lower bound the probability that $\mathcal{A}$'s guess is good, namely, that it "hits" a block on the path that $\mathcal{S}$ chose, and that a collision occurs on that block. First, if $j \neq j_i$ (which means that the block is not on the correct path) then $\mathcal{A}$ fails in the security game. We now prove that with probability $\eta(n_{in})$, there exists a layer in which there is a collision between p and p'.

We denote by i^* the first layer in which a collision occurs, and if no such layer exists (i.e., with probability $1 - \eta(n_{in})$) then $i^* = \perp$. In order to find this i^*, and by that prove its existence, we scan the paths layer by layer, from the top layer 1 to the bottom layer $\ell + 1$. We notice that if $\mathcal{S}$ outputs a path that is a *second opening for* q, then $p_{\ell+1} \neq c_{(\ell+1,q')}$. Thus, it must be that $p \neq p'$, because their last block is different, which implies that there exists i^* such that $p_{i^*} \neq c_{(i^*,j_{i^*})}$ but $h_{i^*-1}(p_{i^*}) = h_{i^*-1}(c_{(i^*,j_{i^*})})$. These are two different strings that are mapped to the same image under h_{i^*-1}.

We turn to prove that if i^* exists, then it is independent of the guess i that $\mathcal{A}$ made. We observe that if $h, h_1, h_2, ..., h_\ell$ are all sampled uniformly at random from $\mathcal{H}$, then

$$\forall i \in [\ell], (h_1, h_2, ..., h_{i-2}, h, h_i, ..., h_\ell) \equiv (h_1, h_2, ..., h_{i-2}, h_{i-1}, h_i, ..., h_\ell)$$

This means that the internal behavior of $\mathcal{S}$, as well as its output, is independent of i, because all of its possible inputs $(h_1, h_2, ..., h_{i-2}, h, h_i, ..., h_\ell)_{i\in[\ell]}$ are sampled from the same distribution. Thus, i and i^* are independent.

Together, we get that the probability that $\mathcal{A}$ breaks the first property (1) of $\mathcal{H}$ is:

$$\Pr_{\mathcal{A} \text{ coins, } h}[\mathcal{A}(h, x^{(1)}) = x^{(2)}, h(x^{(1)}) = h(x^{(2)}), x^{(2)} \neq x^{(1)}] \geq \Pr[i^* \text{ exists} \wedge (i,j) = (i^*, j_{i^*})] \geq \frac{\eta(n_{in})}{N},$$

where the last inequality follows from the fact that $\mathcal{A}$ chooses (i, j) uniformly at random. Since the number of blocks N is upper bounded by the number of nodes, which in turn is bounded by $2L$ (as the tree is a full t-ary tree for $t = n_{in}/n_{out}$, and $t > 2$), we get that $\mathcal{A}$'s success probability is at least

$$\frac{\eta(n_{in})}{2L} = \frac{\eta(n_{in})}{2N/n_{out}}.$$

Since n_{in} and N are polynomially related, and n_{in} and n_{out} as well, N/n_{out} is non-negligible in n_{in}. This implies that $\eta(n_{in})/2L$ is a non-negligible function in n_{in}, and we get that $\mathcal{A}$ breaks the security of $\mathcal{H}$.

Efficiency. The sampling is done efficiently thanks to $\mathcal{H}$'s third property (3). Building T takes $\mathsf{poly}(N, n_{in}, \log(|\Sigma|)) \cdot \ell$ thanks to $\mathcal{H}$'s second property (2), and finding paths in T is also done efficiently. Moreover, $\mathcal{S}$ was assumed to be a PPTM, and computing τ is equivalent to performing $O(\log N)$ operations of integers in $[L]$. The number of layers ℓ is upper bounded by $\log L$, since the arity of the tree is bigger than 2 as noted above. Overall, $\mathcal{A}$ operates in $\mathsf{poly}(N, n_{in}, \log(|\Sigma|))$ time, and thus forms a PPTM. □

Remark 3.7 (The alphabet of the UOWHF tree)**.** *As already mentioned in Footnote 9, in this work we will only use* $\Sigma = \mathbb{GF}[2]$*. However, the strings that we want hash — that is, to use as leaves for the UOWHF tree — are going to be over some finite field* $\mathbb{F}$ *(in particular, we always hash* low degree extensions*, defined next). Thus, given some* $z \in \mathbb{F}^N$ *that we wish to hash, the leaves are actually going to be* $[z]_2 \in \{0,1\}^{N \cdot \log(|\mathbb{F}|)}$*, and this detail is going to be implicit in what follows.*

In this setting, a valid opening may consist of sending $\log(|\mathbb{F}|)$ *paths, one per each bit in the binary representation of the leaf index (the one to be opened). However, since these bits are* consecutive *in* $[z]_2$*, and since a block size* n_{in} *will be bigger than the binary representation of an index (namely,* $n_{in} > \log(|\mathbb{F}|)$*), an opening will only consist up to two paths. To facilitate the reading, we also omit this detail, and only refer to a single path per opening, and to leaf indices in* $[N]$*.*

3.2 Multivariate Polynomials and Low Degree Extensions

We recall some important facts on multivariate polynomials (see [Sud95] for a far more detailed introduction). A basic fact, captured by the Schwartz-Zippel lemma is that low degree polynomials cannot have too many roots.

Lemma 3.8 (Schwartz-Zippel Lemma)**.** *Let* $P : \mathbb{F}^m \to \mathbb{F}$ *be a* non-zero *polynomial of total degree* d*. Then,*

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

An immediate corollary of the Schwartz-Zippel Lemma is that two distinct polynomials $P, Q : \mathbb{F}^m \to \mathbb{F}$ of total degree d may agree on at most a $\frac{d}{|\mathbb{F}|}$-fraction of their domain $\mathbb{F}^m$.

Throughout this work, we consider fields in which operations can be implemented efficiently (i.e., in poly-logarithmic time in the field size). Formally we define such fields as follows.

Definition 3.9. *We say that an ensemble of finite fields* $\mathbb{F} = (\mathbb{F}_n)_{n \in \mathbb{N}}$ *is* constructible *if elements in* $\mathbb{F}_n$ *can be represented by* $O(\log(|\mathbb{F}_n|))$ *bits and field operations (i.e., addition, subtraction, multiplication, inversion and sampling random elements) can all be performed in* $\mathsf{polylog}(|\mathbb{F}_n|)$ *time given this representation.*

A well known fact is that for every $S = S(n)$, there exists a *constructible* field ensemble of size $O(S)$ and its representation can be found in $\mathsf{polylog}(S)$ time using a randomized algorithm (see, e.g., [Gol08, Appendix G.3] for details). Furthermore, if the ensemble extends $\mathbb{GF}[2]$, then the representation can be found deterministically within this time.

Low Degree Extension. Let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) finite fields. Fix an integer $m \in \mathbb{N}$. A basic fact is that for every function $\phi : \mathbb{H}^m \to \mathbb{F}$, there exists a unique extension of ϕ into a function $\hat{\phi} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with ϕ on $\mathbb{H}^m$; i.e., $\hat{\phi}|_{\mathbb{H}^m} \equiv \phi$), such that $\hat{\phi}$ is an m-variate polynomial of individual degree at most $(|\mathbb{H}| - 1)$. Moreover, there exists a collection of $|\mathbb{H}|^m$ functions $\{\hat{\tau}_x\}_{x \in \mathbb{H}^m}$ such that each $\hat{\tau}_x : \mathbb{F}^m \to \mathbb{F}$ is the m-variate polynomial of degree $(|\mathbb{H}| - 1)$ in each variable defined as

$$\hat{\tau}_x(z) \stackrel{\text{def}}{=} \prod_{i \in [m]} \prod_{h \in \mathbb{H} \setminus \{x_i\}} \frac{z_i - h}{x_i - h}, \tag{2}$$

and for every function $\phi : \mathbb{H}^m \to \mathbb{F}$ it holds that

$$\hat{\phi}(z_1, ..., z_m) = \sum_{x \in \mathbb{H}^m} \hat{\tau}_x(z_1, ..., z_m) \cdot \phi(x). \tag{3}$$

The function $\hat{\phi}$ is called the *low degree extension* of ϕ (with respect to $\mathbb{F}$, $\mathbb{H}$ and m).

The low degree extension can be viewed as an error correcting code applied to bit strings. Formally, fix some canonical ordering of the elements in $\mathbb{H}$. For every integer $n \in \mathbb{N}$, we identify the set $[n]$ with the set $\mathbb{H}^{\log_{|\mathbb{H}|}(n)}$ by taking the representation of $i \in [n]$ in base $|\mathbb{H}|$. Consider the function $\mathsf{LDE}_{\mathbb{F},\mathbb{H}} : \{0,1\}^n \to \mathbb{F}^{|\mathbb{F}|^m}$, where $m = \log_{|\mathbb{H}|}(n)$,[11] that given a string $\phi \in \{0,1\}^n$, views ϕ as a function $\phi : \mathbb{H}^m \to \{0,1\}$, by identifying $[n]$ with $\mathbb{H}^m$ as above, and outputs the truth table of the low degree extension $\hat{\phi} : \mathbb{F}^m \to \mathbb{F}$ of ϕ, represented as an $|\mathbb{F}|^m$ dimensional vector.

We use the notation $\mathsf{LDE}(\phi) = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}(\phi)$ or $\hat{\phi}$ to denote the unique low degree extension (with degree $(|\mathbb{H}| - 1)$ in each variable) for inputs of varying lengths. That is, $m = m(n)$ is implicitly taken to be the appropriate number of variables required for any input of length n and a field of size $|\mathbb{H}|$. Typically, m will be clear from the context and otherwise we define it. Once $|\mathbb{H}|$ is set, there is a unique m for every input length, therefore it is well defined.

Proposition 3.10. *Let $\mathbb{H} \subseteq \mathbb{F}$ be constructible field ensembles. There exists a Turing machine that on input $x \in \mathbb{H}^m$, runs in time $O(m \cdot |\mathbb{H}|^2)$ and outputs the polynomial $\hat{\tau}_x : \mathbb{F}^m \to \mathbb{F}$ defined above, represented as an arithmetic circuit over $\mathbb{F}$.*

Moreover, the arithmetic circuit $\hat{\tau}_x$ is an $O(\log n)$-highly uniform circuit (see Definition 3.17) of size $O(m \cdot |\mathbb{H}|^2)$, and there exists a Turing machine with the above time bound that given an input pair $(x, z) \in \mathbb{H}^m \times \mathbb{F}^m$ outputs $\hat{\tau}_x(z)$.

Proof Sketch. The proof follows from the definition of $\hat{\tau}_x$ in Equation (2), and the fact that $\mathbb{H}$ and $\mathbb{F}$ are constructible. First, let us assume that $\mathbb{H}$ is an extension field of $\mathbb{GF}[2]$. There exists an arithmetic $O(|\mathbb{H}|)$-size circuit whose input is two field elements and output is 0 if they are equal and 1 otherwise. For a fixed $i \in [m]$, we locate "in parallel" $|\mathbb{H}|$ copies of this circuit to get a circuit that compares $x_i \in \mathbb{H}$ to any of $\mathbb{H}$'s elements. Then, using m multiplication gates with fan-in 2 that are connected to gates computing $(z_i - h)_{h \in \mathbb{H}}$ for $z_i \in \mathbb{F}$ (notice that, due to our additional assumption, addition and subtraction are equivalent over $\mathbb{F}$) and to the output of these $|\mathbb{H}|$ copies, we get an $O(m \cdot |\mathbb{H}|)$-size circuit that computes the product

$$\prod_{h \in \mathbb{H} \setminus \{x_i\}} (z_i - h).$$

Observe that for every $i \in [m]$,

$$\prod_{h \in \mathbb{H} \setminus \{x_i\}} (x_i - h)^{-1} = \prod_{h \in \mathbb{H} \setminus \{0\}} h,$$

namely, the product of all invertible (non-zero) elements is also the product of the inverses of all invertible elements. Since the latter can be computed in size $\mathbb{H}$, these m gates (one per each $i \in [m]$) can be computed in size $(m \cdot |\mathbb{H}|)$. Overall, the size of the circuit is $O(m \cdot |\mathbb{H}|^2)$. Its uniformity follows from the same arguments used in the proof of Claim 3.22. In order to get rid of the assumption that $\mathbb{H}$ extends $\mathbb{GF}[2]$, one has to replace each subtraction gate with two gates: the first is a multiplication gate that computes the additive inverse of the second term, by multiplying it with the additive inverse of 1, and the second is an addition gate that sums this result with the first term. Notice that this transformation can only increase the circuit's size or depth by a constant factor, and does not hurt its uniformity. □

[11] We assume n is a power of $|\mathbb{H}|$, otherwise we can pad the input with zeros.

Proposition 3.11. *Let $\mathbb{H} \subseteq \mathbb{F}$ be constructible field ensembles. Let $\phi : \mathbb{H}^m \to \mathbb{F}$ and suppose that ϕ can be evaluated by a Turing machine in time t. Then, there exists a Turing machine that, given as an input a point $z \in \mathbb{F}^m$, runs in time $|\mathbb{H}|^m \cdot O(m \cdot |\mathbb{H}|^2) + O(t))$ and outputs the value $\hat{\phi}(z)$ where $\hat{\phi}$ is the unique low degree extension of ϕ (with respect to $\mathbb{H}, \mathbb{F}, m$).*

Proof. The Turing machine computes

$$\hat{\phi}(z) = \sum_{x \in \mathbb{H}^m} \hat{\tau}_x(z) \cdot \phi(x)$$

by generating and evaluating $\hat{\tau}_x(z)$ as in Proposition 3.10. □

In certain cases, we need to split a claim about the low degree extension of multiple strings into multiple claims about the low degree extension of each one of them. A natural way of doing so is to leverage the fact that the low degree extension is a tensor code.

In detail, given strings $\phi_1, ..., \phi_k$ of varying lengths, denote by ϕ' the longest one and denote its length by n'. Take m such that $|\mathbb{H}|^m \geq n'$ and place all inputs in an $|\mathbb{H}| \times |\mathbb{H}|^m$ matrix, where the i^{th} row contains ϕ_i and empty cells are filled with zeros.[12] As before, interpret every row of the matrix as a function $\mathbb{H}^m \to \mathbb{F}$ (an m-variate polynomial of degree-$(|\mathbb{H}| - 1)$ in each variable), and each column as a function $\mathbb{H} \to \mathbb{F}$ (a degree-$(|\mathbb{H}| - 1)$ univariate polynomial). Apply the low degree extension on each row and each column to get an $|\mathbb{F}| \times |\mathbb{F}|^m$ matrix.

Now, the probability that a uniformly random element of the matrix $r' \in \mathbb{F}^{m+1}$ resides in the first $|\mathbb{H}|$ rows is $|\mathbb{H}|/|\mathbb{F}|$. Otherwise, the value at r' depends on all low degree extensions of $\phi_1, ..., \phi_k$: It is a linear combination of them and one may split a claim about the value of $\mathsf{LDE}(\phi_1, ..., \phi_k)[r']$ to k claims about $\hat{\phi}_1, ..., \hat{\phi}_k$, using Equation (3). Namely, there exists coefficients $\alpha_1, ..., \alpha_k \in \mathbb{F}$ and a coordinate $r \in \mathbb{F}^m$ such that

$$\mathsf{LDE}(\phi_1, ..., \phi_k)[r'] = \alpha_1 \cdot \mathsf{LDE}(\phi_1)[r] + ... + \alpha_k \cdot \mathsf{LDE}(\phi_k)[r]. \tag{4}$$

The following proposition shows how to use this property in the interactive setting: namely, that there exists an interactive reduction from the $(m+1)$-dimensional claim to k m-dimensional claims that outputs their k coefficients as described in Equation (4).

Proposition 3.12. *Let $(r', v') \in \mathbb{F}^{m+1} \times \mathbb{F}$ be a claim about the low degree extension $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ of $k \leq |\mathbb{H}|$ strings $(\phi_1, ..., \phi_k)$, namely, $\mathsf{LDE}(\phi_1, ..., \phi_k)[r'] = v'$. Set r to be the projection of r' on $\{0\} \times \mathbb{F}^m$. Then, there exists an interactive protocol between a prover and the verifier, such that the verifier outputs k values $v_1, ..., v_k \in \mathbb{F}$ and k coefficients $\alpha_1, ..., \alpha_k \in \mathbb{F}$ that satisfy*

$$\mathsf{LDE}(\phi_1, ..., \phi_k)[r'] = v' \iff \bigwedge_{i=1}^{k} (\alpha_i \cdot \mathsf{LDE}(\phi_i)[r] = v_i).$$

The verifier runs in time $O(|\mathbb{H}|^2 \cdot \sum_{i=1}^{k} m(|\phi_i|))$, the prover runs in time $O(|\mathbb{H}|^2 \cdot \sum_{i=1}^{k} m(|\phi_i|) \cdot |\phi_i|)$, the communication complexity is $k \cdot \log(|\mathbb{F}|)$ and the number of rounds is $1/2$ (i.e., a single message).

[12] Actually, it may be the case that $k > |\mathbb{H}|$, and then we need to perform the same process more than one time, i.e., where the elements of the matrix are vectors or matrices themselves, instead of field elements. In our usages, however, it always holds that $k < |\mathbb{H}|$.

Proof. The prover computes $v_i = \mathsf{LDE}(\phi_i)[r]$, for every $i \in [k]$, and sends these values to the verifier. This takes $O(|\mathbb{H}|^2 \cdot \sum_{i=1}^{k} m(|\phi_i|) \cdot |\phi_i|)$ time. The verifier checks that $v' = \sum_{i=1}^{k} v_i$, and otherwise rejects. If this test passed, it outputs the received values $v_1, ..., v_k$, together with the coefficients $\alpha_1, ..., \alpha_k \in \mathbb{F}$ that he computes. Notice that the i^{th} coefficient α_i can be found in time $O(m(|\phi_i|) \cdot |\mathbb{H}|^2)$ due to Proposition 3.10.

Soundness follows from the fact if the original claim (r', v') was false, then either at least one of the alleged claims (r, v_i) is false, or the sum of $v_1, ..., v_k$ is not equal to v'. □

We comment that it may be that only one of these coefficients is non-zero, in the case that $r'[1] \leq |\mathbb{H}|$. The aforementioned reduction is still correct, but we have not gained anything. We further comment that if the verifier is able to compute one of the values v_i on its own, it can certainly do so and subtract the result from v' (instead of subtracting the alleged value for v_i that the prover provides). Of course, this change does not hurt the soundness of the reduction.

Next, the following claim implements the standard interactive process of reducing the task of proving many claims about the low degree extension of a string to the task of proving a single claim about it. 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 outputs a random point on the curve and its alleged value. Soundness follows from the fact that composing a low degree curve with a low degree polynomial results in a low degree univariate polynomial.

Claim 3.13 (Reducing the number of claims)**.** *Fix some input w and a set of t claims, denoted $(\chi_i, \theta_i)_{i \in [t]} \in (\mathbb{F}^m \times \mathbb{F})^t$, about $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(w) = \hat{w}$. Then, for every $\varepsilon \in (0, 1]$, there exists an interactive protocol that outputs a single claim (χ, θ) such that:*

- **Completeness.** *If $\forall i \in [t], \hat{w}[\chi_i] = \theta_i$, then $\hat{w}[\chi] = \theta$.*
- **Soundness.** *If $\exists i \in [t]$ such that $\hat{w}[\chi_i] \neq \theta_i$, then, for every prover strategy, with probability $1 - (m \cdot |\mathbb{H}| \cdot t^{\varepsilon})/(\varepsilon \cdot |\mathbb{F}|)$ over the verifier coin tosses, it holds that $\hat{w}[\chi] \neq \theta$.*

The prover runs in time $\mathsf{poly}(|\mathbb{F}|^m)$, the verifier runs in time $((1/\varepsilon) \cdot m \cdot |\mathbb{H}| \cdot t^{1+\varepsilon} \cdot \log(|\mathbb{F}|))$, the communication complexity is $((1/\varepsilon) \cdot m \cdot |\mathbb{H}| \cdot t^{1+\varepsilon} \cdot \log(|\mathbb{F}|))$ and the number of rounds is $1/\varepsilon$.

Proof. We split the set $(\chi_i, \theta_i)_{i \in [t]}$ into $t^{1-\varepsilon}$ "batches" of size at most t^{ε} each. We run the following process on each of the batches, and repeat applying it on the results for $1/\varepsilon$ iterations:

1. Let $(\rho_i)_{i \in [t^{\varepsilon}]} \in \mathbb{F}^{t^{\varepsilon}}$ be some canonical set of distinct fixed elements known to the prover and to the verifier. For each batch $(\chi_i, \theta_i)_{i \in [t^{\varepsilon}]}$, let $\gamma : \mathbb{F} \to \mathbb{F}^m$ be the unique degree-$(t^{\varepsilon} - 1)$ curve that passes through the batch of points $(\chi_i)_{i \in [t^{\varepsilon}]}$, such that $\forall i, \gamma(\rho_i) = \chi_i$. The prover sends the function $\hat{w} \circ \gamma : \mathbb{F} \to \mathbb{F}$ to the verifier.
2. Upon receiving a function $f : \mathbb{F} \to \mathbb{F}$ from the prover (supposedly, $f = \hat{w} \circ \gamma$), the verifier checks that f is a polynomial of degree $m \cdot (|\mathbb{H}| - 1) \cdot (t^{\varepsilon} - 1)$, and that $\forall i, f(\rho_i) = \theta_i$. If these tests pass, then the verifier chooses a random element $\rho \in \mathbb{F}$ and sends it to the prover.
3. The prover and the verifier continue to the next iteration, such that the batch $(\chi_i, \theta_i)_{i \in [t^{\varepsilon}]}$ was reduced to the single claim $(\gamma(\rho), f(\rho))$.

Prescribed completeness is trivial. For soundness, imagine a t^ε-ary tree, where each internal node represents a claim that is the result of the following process when applied to its children. The leaves represent the original claims $(\chi_i, \theta_i)_{i\in[t]}$. Assume that the i^{th} claim is false, i.e., that $\hat{w}[\chi_i] \neq \theta_i$, and assume that for some cheating prover strategy, the probability that the output claim is correct is s. That is,

$$s = \Pr[\hat{w}[\chi] = \theta].$$

Observe that the i^{th} leaf represents a false claim but the root represents a correct claim, and consider the unique path that connects the root and the i^{th} leaf. There must exist some node in this path that represents a correct claim, but at least one of its children represents a false claim. We focus on the interactive process that is applied to these children. In this process, the only way for the prover to pass the tests in step (2) is by sending a function f such that $f \neq \hat{w} \circ \gamma$ but $f(\rho) = (\hat{w} \circ \gamma)(\rho)$. This implies that f and $\hat{w} \circ \gamma$ are two distinct polynomials of degree at most $m \cdot (|\mathbb{H}| - 1) \cdot (t^\varepsilon - 1)$, and thus by the Schwartz-Zippel Lemma (see 3.8), we get that

$$\Pr[f(\rho) = \hat{w} \circ \gamma(\rho)] \leq \frac{m \cdot (|\mathbb{H}| - 1) \cdot (t^\varepsilon - 1)}{|\mathbb{F}|} \leq \frac{m \cdot |\mathbb{H}| \cdot t^\varepsilon}{|\mathbb{F}|}.$$

By a Union Bound over all $1/\varepsilon$ nodes in the path, we get that

$$s \leq \frac{m \cdot |\mathbb{H}| \cdot t^\varepsilon}{\varepsilon \cdot |\mathbb{F}|}.$$

Notice that sending or evaluating a polynomial over $\mathbb{F}$ with (total) degree at most $m \cdot (|\mathbb{H}| - 1) \cdot (t^\varepsilon - 1)$ takes $m \cdot (|\mathbb{H}| - 1) \cdot (t^\varepsilon - 1) \cdot \log(|\mathbb{F}|) < m \cdot |\mathbb{H}| \cdot t^\varepsilon \cdot \log(|\mathbb{F}|)$ bits or time. To conclude, observe that there are at most $t/(t^\varepsilon - 1) \leq t/\varepsilon$ nodes in three and that there are $2/\varepsilon$ messages, and the other complexity measures follow by construction. □

3.3 Circuit Classes, Uniformity and Succinct Descriptions

Ensembles of Boolean and arithmetic circuits play an important role in many of our protocols. Throughout this work, circuit ensembles $\{C_n\}_{n=1}^\infty$ are indexed by an integer n. We *do not (necessarily) take n to be the n-th circuit's input length*: an ensemble is an arbitrary sequence of circuits of *non-decreasing* input lengths. We usually refer to the input length by $n_{input} = n_{input}(n)$ (where the case $n_{input} = n$, which is the more common use in the literature, is a special case). We can refer to the language computed by a circuit ensemble, which is the set $\{(n, x) : x \in \{0,1\}^{n_{input}(n)} \bigwedge C_n(x) = 1\}_{n=1}^\infty$. We measure circuit complexity as a function of the index n (rather than the input length). For example, in an ensemble of polynomial-size circuits, the n-th circuit C_n can have size $\mathsf{poly}(n)$ even if the input x is say of length $O(\log(n))$. Separating the index n from the input length allows us to tie together several different circuits of varying input lengths that are used in our protocol. Throughout our work, the circuit's input length will be at most polynomial in n (but it can be smaller than n, even logarithmic), i.e. $n_{input}(n) \leq \mathsf{poly}(n)$.

For pair languages (or triplet languages), we take the lengths of the various inputs to also be a function of n. For example, for inputs of the form (x, y), where x is an explicit input and y is an implicit input, we have $n_{input}(n) = |x| + |y| = n_{exp}(n) + n_{imp}(n)$.

Boolean and arithmetic circuits. Throughout this work we refer to several circuit classes, and the languages they can compute. AC^0 circuits are ensembles of Boolean circuits with polynomial size, constant depth, and unbounded fan-in. $\mathsf{AC}^0[\oplus]$ circuits also allow parity gates of unbounded fan-in. NC^1 circuits are Boolean circuits of logarithmic depth and constant fan-in. Probabilistic circuits are also allowed to use random coins as part of their input. We also refer to the corresponding complexity classes of languages (or functions) computable by circuits of each type (i.e. the classes $\mathsf{AC}, \mathsf{AC}^0[\oplus], \mathsf{NC}^1$), where in defining the complexity classes we restrict to ensembles where the input length is $n_{input} = n$.

We make extensive use of arithmetic circuits over a field $\mathbb{F}$: $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits are constant-depth arithmetic circuits, with addition and multiplication gates over the field $\mathbb{F}$, where addition gates have unbounded fan-in, but the fan-in of multiplication gates is bounded by $f_{in}(n)$ (typically n^σ for a small constant σ). Probabilistic arithmetic circuits are allowed to use random coins as part of their input.[13] We also refer to the corresponding complexity class $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ of arithmetic functions computable by such circuits with input length $n_{input} = n$.

We sometimes use arithmetic circuits to compute Boolean functions:

Definition 3.14 (Arithmetic circuit computing a Boolean function)**.** *An arithmetic circuit $\widehat{C}$ on n_{input} inputs computes a Boolean function $f : \{0,1\}^{n_{input}} \to \{0,1\}$ if for every Boolean input, the circuit outputs 1 if and only if $f(x) = 1$. If $f(x) = 0$, then we allow $\widehat{C}$ to output any field element that is not 1 (we remark that in some of the arithmetic circuits we use to compute Boolean functions, it will be the case that they output 0 when $f(x) = 0$, but this will not always be the case).*

We say that a circuit ensemble $\{\widehat{C}_n\}$ computes an ensemble of Boolean functions $\{f_n\}$ if the above condition holds for every integer n.

Uniformity and succinct descriptions. Circuit uniformity plays an important role in our protocols: specifically, in a protocol for verifying the output of a large circuit, the verifier (whose running time is smaller than the circuit size) needs to have a succinct implicit description of the circuit. There are different notions of uniformity that have been studied in this context. The most flexible notion of uniformity that we consider in this work is *log-space uniformity*, where there should exist a log-space Turing machine that, on input 1^n, outputs the entire circuit C_n.

For several of our protocols, we need to utilize more fine-grained or restrictive uniformity notions. Following the formalization in [GR20], we consider the complexity of a succinct (implicit) representation of the circuit. We consider two such representations:

- An *adjacency predicate* that indicates, for a pair of gates u and v, whether gate u is fed by gate v.
- An *incidence function* that indicates, for a gate u and an index i, the identity of the i-th gate that feeds gate u (by convention, the value 0 indicates that the gate has fewer than i gates).

Notice that when these representations are Boolean (rather than arithmetic), they are referred to as Boolean formulas, which extends the standard definition of a Boolean formula, since the incidence function is vector-valued: We refer to any Boolean circuit as a Boolean formula as long as each of its internal gates have fan-out 1, or, in other words, if each of its output bits can be computed by a (standard) Boolean formula.

[13] Unless otherwise noted, we assume the random inputs to a randomized arithmetic circuit are each drawn uniformly and i.i.d. from $\{0,1\}$.

In all cases, we assume that the circuit (Boolean or arithmetic) is *layered*: gates at layer $(i-1)$ are fed only by gates in layer i, where layer i consists of all gates at distance i from the output gate(s). This also means that each layer contains all of the constant gates (which are moved from layer to layer), rather than only the input layer: in case of Boolean circuits, these are 0 and 1, and in case of arithmetic circuits, these are all of the elements of $\mathbb{F}$. Following [GR20], and without loss of generality, we make several simplifying assumptions. For Boolean circuits, we assume that for each layer i, all gates in that layer have the same functionality. For $\mathsf{AC}^0[\oplus]$ circuits, we assume that there are no negation gates (only addition and parity gates). Moreover, for $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits, we allow addition and multiplication gates, and we assume that for each layer i, all gates are of the same type (addition or multiplication). See [GR20] for a more thorough discussion of these assumptions and why they can be made without loss of generality. Finally, we assume that claims about the LDE of a circuit's input layer are only about the actual input to the circuit, rather than on the constant gates as well. In all of the interactive protocols that we use in this work, this assumption can be justified using Proposition 3.12 (in particular, see the comment after the proposition: the verifier can compute on it own the values of the low-degree extension that correspond to the constant gates). The cost in the complexity measures will be immaterial in all cases.

We say that a circuit is highly uniform if the appropriate function (adjacency predicate or incidence function) can be represented by a sufficiently small formula, where this formula itself can be constructed (and evaluated) by a Turing machine whose running time is bounded. In some of our results we use circuits where constructing the above formula requires a short advice string. A circuit is *s-succinct* (and highly uniform) if there is an s-bit advice string s.t. when this advice string is fed to a Turing machine it outputs the formula computing the adjacency predicate or incidence function. Intuitively, the verifier in an interactive protocol will need to run this Turing machine (with the appropriate advice string) to compute the formula and evaluate it on various inputs.

For Boolean circuits, we only require a formula as above for computing the adjacency predicate:

Definition 3.15 (Succinct and highly uniform Boolean circuit)**.** *An ensemble $\{C_n\}$ of Boolean circuits is* $s(n)$-succinct and highly uniform *if there exists a Turing machine M and an ensemble $\{a_n\}$ of $s(n)$-bit advice strings, such that on input (n, a_n) the Turing machine M runs in time $(s^{1+o(1)} \cdot n^{o(1)})$ and outputs a formula of size $(s^{1+o(1)} \cdot n^{o(1)})$ that computes the circuit C_n's adjacency predicate.*

If $s(n) = 0$, i.e. the advice is empty, we simply say that $\{C_n\}$ is highly uniform*.*

For arithmetic circuits, we require a formula as above for the adjacency predicate of addition gates. For multiplication gates, we need a formula for computing the incidence function. We emphasize that even though the circuits are arithmetic, the adjacency predicate and the incidence function are Boolean formulas (the incidence function is a vector-valued Boolean function).

Definition 3.16 (Succinct and highly uniform arithmetic circuits)**.** *An ensemble $\{C_n\}$ of arithmetic circuits over fields $\{\mathbb{F}_n\}$ is* $s(n)$-succinct and highly uniform *if there exists a Turing machine M and an ensemble $\{a_n\}$ of $s(n)$-bit advice strings, such that on input (n, a_n) the Turing machine M runs in time $(s^{1+o(1)} \cdot n^{o(1)})$ and outputs two Boolean formulas of size $(s^{1+o(1)} \cdot n^{o(1)})$. The first formula computes the adjacency predicate for the circuit C_n's addition gates. The second formula computes the incidence function for the circuit C_n's multiplication gates.*

If $s(n) = 0$, i.e. the advice is empty, we simply say that $\{C_n\}$ is highly uniform*.*

In addition to bounding the *size* of the Boolean formulas that compute the circuit's adjacency predicate or incidence function, one may also want to bound their *degree*, when viewing them as arithmetic circuits over $\mathbb{GF}[2]$. Notice that an arithmetic representation of these functions is not necessarily the result of a direct arithmetization of a Boolean formula; rather, any small arithmetic circuit over $\mathbb{GF}[2]$ that computes these functions can satisfy the definition. This alternative requirement will be useful in bounding the soundness error of the HIPP of Theorem 3.26.

Definition 3.17 (Succinct and $d(n)$-highly uniform circuits)**.** *An ensemble $\{C_n\}$ of Boolean (resp., arithmetic) circuits is* $s(n)$-succinct and $d(n)$-highly uniform *if there exists a Turing machine M and an ensemble $\{a_n\}$ of $s(n)$-bit advice strings, such that on input (n, a_n) the Turing machine M runs in time $(s^{1+o(1)} \cdot n^{o(1)})$ and outputs an arithmetic circuit over $\mathbb{GF}[2]$, of size $(s^{1+o(1)} \cdot n^{o(1)})$ and degree at most $d(n)$, that computes the circuit C_n's adjacency predicate (resp., another circuit for computing C_n's incidence function, analogously to Definition 3.16).*

Using the approximation method of Razborov [Raz87] and Smolensky [Smo87], Goldreich and Rothblum [GR20] show that highly uniform Boolean $\mathsf{AC}^0[\oplus]$ circuits can be approximated by highly-uniform probabilistic arithmetic $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits:

Theorem 3.18 (Approximating $\mathsf{AC}^0[\oplus]$ circuits via $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ [GR20])**.** *Let $\mathcal{L}$ be a language that is computable by an ensemble of $s(n)$-succinct and $d(n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuits $\{C_n\}$. Then for any field $\mathbb{F}$ and integer-valued function $\lambda(n) \leq n$, there exists an ensemble of $s(n)$-succinct and $d(n)$-highly uniform* probabilistic $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ *circuits $\{\widehat{C}_n\}$ of size $(|C_n| \cdot \lambda(n) \cdot n^{o(1)})$ where $f_{in} = O(\log(|C_n|) + \lambda(n))$ and for every input $x \in \{0,1\}^n$:*

$$\Pr_{\widehat{C}_n\text{'s coins}}\left[\widehat{C}_n(x) \neq C_n(x)\right] \leq 2^{-\lambda(n)}.$$

The number of random coins used by the circuit $\widehat{C}_n$ is $O((\lambda(n) + \log|C_n|) \cdot \log(n))$. We refer to these random coins as the seed.

We remark that here the approximation is strong, in the sense that the arithmetic circuit's output agrees with the Boolean output w.h.p. (thus, it is stronger than our default notion of Definition 3.14, because it always outputs 0 when $C_n(x) = 0$ instead of an arbitrary field element that is not 1). On the other hand, it is weaker than an arithmetic circuit that *computes* a Boolean function, as the approximation is probabilistic.

We also emphasize that the fan-in of the multiplication gates in the resulting circuits is logarithmic (in the size of the circuit and the inverse of the error probability). This is in contrast to the straightforward arithmetization (replacing AND gates with multiplications), where the fan-in would be polynomial. Another important aspect of this theorem is that even though we do not have a small and low-degree arithmetic circuit for the *incidence function* of the $\mathsf{AC}^0[\oplus]$ circuit, we can get such a small circuit for computing the incidence function of the $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit's multiplication gates. Moreover, the degree of the adjacency predicate for the circuit's addition gates is not increased by more than a logarithmic factor under this transformation.

Remark 3.19. *In the interactive proof setting, the choice of the seed when using the foregoing theorem is made by the verifier: It uniformly selects it and sends it to the prover. Moreover, in order to get rid of the completeness error that this randomized reduction carries, at the end of the interaction the prover needs to send a message that indicates if the random string was "good" (and*

convince the verifier otherwise in case it is not). This only adds a single round of communication (a single verifier message and a single prover message), whereas it increases the communication only by $O((\lambda(n) + \log|C_n|) \cdot \log(n))$ bits.

Succinct functions and sets. We next define a notion of succinct representation of functions. Loosely speaking, a function f, which can be Boolean or arithmetic over a field $\mathbb{F}$, has an s-succinct d-highly uniform representation if it can be computed by circuits in a low complexity class that are also s-succinct and d-highly uniform circuits (see Definitions 3.15, 3.16 and 3.17). In particular, this means that there is an s-bit string $\langle f \rangle$ that describes f. The actual definition requires that the circuit computing f reside in $\mathsf{AC}^0[\oplus]$ (for Boolean functions) or $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ (for arithmetic functions):

Definition 3.20 (Succinct Description of Functions)**.** *A Boolean function f has an $s(n)$-*succinct *$d(n)$-*highly uniform description *if it can be computed by an $s(n)$-succinct and $d(n)$-highly uniform ensemble of $\mathsf{AC}^0[\oplus]$ circuits. We use $\langle f \rangle$ to refer to the succinct description of f (the advice string in Definition 3.15).*

*An arithmetic function f has an $s(n)$-*succinct *$d(n)$-*highly uniform description *if it can be computed by an $s(n)$-succinct and $d(n)$-highly uniform ensemble of $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits. We use $\langle f \rangle$ to refer to the succinct description of f (the $s(n)$-bit advice string in Definition 3.16).*

When $d(n)$ is not specified, we say that f has an $s(n)$-succinct description if it can be computed by an $s(n)$-succinct and highly uniform ensemble. Note that by Theorem 3.18, any $s(n)$-succinct Boolean function f can also be viewed as a $s(n)$-succinct arithmetic probabilistic function $\widehat{f}$, where for every input in $\{0,1\}^*$, f and $\widehat{f}$ agree w.h.p.

We also define succinct representation for sets $A \subseteq [n]$. Roughly speaking, this means that there is a succinct function that can be used to efficiently compute a list of A's elements. To facilitate efficient computation and uniformity, the formal definition is somewhat more involved: it allows the list to have the elements of A in arbitrary order and with multiplicities.

Definition 3.21 (Succinct Description of Sets)**.** *An ensemble $\{A_n\}$ of sets has an $s(n)$-succinct $d(n)$-highly uniform description as a list of length $q(n)$ if there exists an ensemble of $s(n)$-succinct and $d(n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuits $\{C_{A,n}\}$, where $C_{A,n}$ maps indices in $[q(n)]$ (given in binary representation) to elements in $[m(n)]$, such that for every value of n, the following holds: Let v_A be the $q(n)$-dimensional vector whose i^{th} entry is the output of $C_{A,n}$ on input i, then the set of distinct elements in v_A exactly equals A_n.*

We refer to $\langle A \rangle$ as the succinct representation of A (the $s(n)$-bit advice string in Definition 3.15). We refer to $[m(n)]$ as A's domain, of size $m(n)$.

As in Definition 3.20, when $d(n)$ is not specified, the ensemble $\{C_{A,n}\}$ should be highly uniform. Note that each element of A must appear at least once in v_A, but if $q > |A|$ then elements of A can appear more than once (indeed, at least one element must appear multiple times). We remark that the circuit mapping indices in $[q]$ to elements in A can be used (in parallel) to compute the entire list. This blows up the circuit size by a multiplicative factor of q (but doesn't increase the depth or worsen the uniformity). Finally, the formula computing the adjacency predicate of the circuit is of size proportional to the description s (which can be much smaller than $|A|$).

Compared to the definition of *succinct description of sets* in [RRR18, Definition 2.2], that definition is more straightforward, as it allows $\mathcal{NC}^1$ circuits. Since we require that the circuit that outputs the elements of A is an $\mathsf{AC}^0[\oplus]$ circuit, we need a more flexible definition.

Binary representation of field elements. Take $\mathbb{F} = \{f_0, ..., f_{|\mathbb{F}|-1}\}$, where $f_0 = 0$ and $f_1 = 1$. If $\mathbb{F}$ is an extension field of $\mathbb{GF}[2]$, then field elements can be represented as polynomials with binary coefficients with degree at most $(\log(|\mathbb{F}|)-1)$. We define the $(\log(|\mathbb{F}|))$-bit vector of coefficients as *the binary representation of an element*, such that each bit in the bit string corresponds to the coefficient in the polynomial at the same position. The following claim shows that this representation can be computed by highly-uniform circuits. Its proof can be found in Appendix A.4.

Claim 3.22. *Let $\mathbb{F}$ be an extension field of $\mathbb{GF}[2]$. The binary representation of a field element can be computed in $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ with multiplication gates of fan-in $|\mathbb{F}|$.*

3.4 HIP and HIPP

In this work, we build several proof-systems of a special type, called a *Holographic Interactive Proof* (HIP) — a notion formally defined in the work of Gur and Rothblum [GR17]. A HIP is defined similarly to a standard interactive proof, except that the verifier, rather than being given access to the main input explicitly, outputs a claim about the encoding of the input. Thus, the input is called *holographic*. The hope is that the redundancy provided by the encoding will allow the verifier to run in sublinear time.

As a matter of fact, it turns out that for some codes (specifically the low degree extension), reading just a single point r from the encoded input suffices for the verifier (we later implement this idea in Claim 3.13). 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 r, the verifier outputs a claim about the point, i.e., it outputs r together with a value v that it would have expected to see, had it actually queried the (encoded) input at r.

We generalize the definition for pair languages. On input (x, z), we interpret x as the explicit input and z as the holographic input. The prover gets them while the verifier only gets $(x, |z|)$. The claim about the encoding of the input is only about the holographic input z.

We focus on HIPs with respect to the low degree extension encoding (which is formally defined in Section 3.2), thus the definition we provide is only for this family of codes.

Definition 3.23 (Holographic Interactive Proofs (HIP))**.** *Fix finite fields $\mathbb{H} \subseteq \mathbb{F}$ and a low degree extension encoding $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$.*

A Holographic Interactive Proofs *for a pair language $\mathcal{L}$, with respect to the low degree extension LDE, is an interactive protocol with two parties: a (computationally unbounded) prover $\mathcal{P}$ and a computationally bounded verifier $\mathcal{V}$. Both parties get as input $x \in \{0,1\}^{n_{exp}}$. The prover also gets $z \in \{0,1\}^{n_{hol}}$ whereas the verifier only gets $|z| = n_{hol}$.*

At the end of the interaction, either the verifier rejects or it outputs a coordinate $r \in \mathbb{F}^{m(n_{hol})}$, and a value $v \in \mathbb{F}$, such that:

- **Completeness.** *If $(x, z) \in \mathcal{L}$ and the prover honestly follows the protocol, then $\mathsf{LDE}(z)[r] = v$.*
- **Soundness.** *If $(x, z) \notin \mathcal{L}$, then for any (unbounded) cheating prover, with probability at least $1/2$ over the verifier's coins, $\mathsf{LDE}(z)[r] \neq v$.*

In the setting of holographic proofs *of proximity*, we consider a triplet language $\mathcal{L}$ with inputs of the form (x, y, z). As in the HIP definition, x is the explicit input and z is the holographic input. The input y is *implicit*; it is an auxiliary information that the prover gets, while the verifier only

gets *query access* to (but never gets explicit access to or outputs a claim about). In the case that $\mathcal{L}$ represents an NP relation, we think of y as the NP witness for the membership of (x, z) in the underlying NP language.

Loosely speaking, we say that an IPP (or a holographic IPP) is oblivious if the verifier makes all its queries to y non-adaptively at the end of the interaction. Put differently, we eliminate its query access to y, and instead the verifier specifies some query set Q of bits from y and a predicate ψ, that captures the conditions that would make it not reject (and to output a claim (r, v) about the encoding of the holographic input, in the holographic case). On input (x, y, z), the probability that the verifier does not reject and outputs ψ such that $\psi(y_Q) = 1$, is related to the distance of y from the set $\mathcal{L}(x, z) = \{y' : (x, y', z) \in \mathcal{L}\}$: if $y \in \mathcal{L}(x, z)$, then this probability is 1, whereas if y is ε-far from $\mathcal{L}(x, z)$, then this probability is at most $1/2$.

For technical considerations in our proof, we actually require that the verifier generates succinct descriptions of Q and ψ (see Definitions 3.20 and 3.21 for the precise technical definition of succinct descriptions of functions and sets). This allows the verifier to run in time that is sublinear in the sizes of Q and ψ. We proceed with the formal general definition, that does not require Q or ψ to be succinct:

Definition 3.24 (Oblivious Holographic IPP)**.** *Fix finite fields $\mathbb{H} \subseteq \mathbb{F}$ and a low degree extension encoding $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$.*

An Oblivious Holographic IPP (oblivious HIPP) *for a triplet language $\mathcal{L}$, with respect to the low degree extension LDE, is an interactive protocol with two parties: a (computationally unbounded) prover $\mathcal{P}$ and a computationally bounded verifier $\mathcal{V}$. Both parties get as input $x \in \{0,1\}^{n_{exp}}$ and a proximity parameter $\varepsilon > 0$. The prover also gets $y \in \{0,1\}^{n_{imp}}$ and $z \in \{0,1\}^{n_{hol}}$ whereas the verifier only gets $(|y| = n_{imp}, |z| = n_{hol})$.*

At the end of the interaction, either the verifier rejects or it outputs: (1) a coordinate $r \in \mathbb{F}^{m(n_{hol})}$ and a value $v \in \mathbb{F}$; (2) a description $\langle Q \rangle$ of a set $Q \subseteq [n_{imp}]$ of size q and a description $\langle \psi \rangle$ of a predicate $\psi : \{0,1\}^q \to \{0,1\}$, such that:

- **Completeness.** *For every triplet $(x, y, z) \in \mathcal{L}$ and proximity parameter $\varepsilon > 0$ it holds that*

$$\Pr[\mathcal{V} \text{ does not reject, } \psi(y_Q) = 1 \text{ and } \mathsf{LDE}(z)[r] = v] = 1.$$

- **Soundness.** *For every $\varepsilon > 0$, $(x, z) \in \{0,1\}^{n_{exp}} \times \{0,1\}^{n_{hol}}$ and every a priori fixed y that is ε-far from the set $\{y' : (x, y', z) \in \mathcal{L}\}$, and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that*

$$\Pr[\mathcal{V} \text{ does not reject, } \psi(y_Q) = 1 \text{ and } \mathsf{LDE}(z)[r] = v] \leq 1/2.$$

The query complexity *of an oblivious holographic IPP is q, the size of the set Q, and the* communication complexity *is the number of bits exchanged between $\mathcal{V}$ and $\mathcal{P}$. We say that the IPP has an efficient prover strategy if the honest prover strategy $\mathcal{P}$ can be implemented in polynomial time.*

In order to prove Lemma 4.6 (see Section 4.4), which is our main technical tool, we utilize an idea from the work of Reingold et al. [RRR18] who used known IPP protocols to achieve batch verification for UP languages. There, they introduced and studied the notion of *Interactive Witness Verification* (IWV). We comment that an IPP for a pair language (resp., Oblivious IPP), in which the holographic input is empty, is a strict generalization of an IWV (resp., Oblivious IWV).

The first difference, as its name suggests, is that an IWV is only defined for an NP relation, rather than for any pair (or triplet) language. In other words, the input satisfies the relation if and only if the implicit input is the NP witness of the explicit input. The second difference is that a general IPP depends on the distance, i.e., the probability that the verifier accepts depends on the distance of the implicit input from the language induced by the explicit and the holographic inputs: if the implicit input is in the language, then this probability is 1, whereas if it is far from it, then this probability is at most 1/2. In contrast, in an IWV, the verifier rejects w.h.p. for any fixed false witness, regardless of its distance from a valid NP witness for the input.

We avoid defining or using IWV protocols, as they did, since our usage requires using the holographic version of these. Thus, instead, we use the more generalized form of holographic IPPs for triplet languages.

3.5 Constant-round HIPs and HIPPs for $\mathsf{AC}^0[\oplus]$ and $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$

Constant-Round HIP. The following result is derived from the constant round interactive proof of Goldreich and Rothblum [GR20].

Theorem 3.25 (Constant-round HIP for $\mathsf{AC}^0[\oplus]$, $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ [GR20])**.** *Let $\delta \in (0,1]$ be a constant, and let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) extension fields of $\mathbb{GF}[2]$, where $|\mathbb{H}_n| = \Theta(n^\delta)$ and $|\mathbb{F}_n| = \mathsf{poly}(n)$. Let $\mathcal{L}$ be a pair language with input length $n_{input} = n_{exp} + n_{hol} \leq \mathsf{poly}(n)$ that can be computed:*

- *either in $s(n)$-succinct and $d(n)$-highly uniform $\mathsf{AC}^0[\oplus]$, with circuit size $O(n^C)$, circuit depth $D = O(1)$, and succinct description length $s(n) \leq n$,*
- *or in $s(n)$-succinct and $d(n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$, with circuit size $O(n^C)$, circuit depth $D = O(1)$, succinct description length $s(n) \leq n$, and multiplication gate fan-in $f_{in} = O(n^\delta)$.*

There is a holographic interactive proof (HIP) for $\mathcal{L}$ with respect to the low degree extension $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ (see Definition 3.23) with the following complexity measures:

- *the number of rounds is $O(D \cdot C/\delta)$,*
- *the communication complexity is $(n^{O(\delta)} \cdot s(n)^{1+o(1)})$,*
- *the verifier, who gets the $s(n)$-bit string describing the circuit as an additional explicit input, runs in time $(n^{O(\delta)} \cdot (s(n)^{1+o(1)} + n_{\mathsf{exp}}))$,*
- *the prover, who gets explicit access to all inputs (including the string describing the circuit) runs in time $\mathsf{poly}(n)$,*
- *the HIP has perfect completeness, and the soundness error is $(n^{O(\delta)} \cdot d/|\mathbb{F}|)$.*

See Appendix A.2 for a review of the construction (there are some technical differences from the protocol and presentation in [GR20]). The result is shown for $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits, that may be vector-valued. In this case, all output wires equal 1 if and only if the input is in the language, and this generalization is possible at the same cost as for a single output (see Remark A.1).

The result for $\mathsf{AC}^0[\oplus]$ circuits follows using the approximation method (Theorem 3.18).

Constant-round HIPP. Our protocols make use of a constant-round holographic oblivious IPP for languages that are computable in highly uniform $\mathsf{AC}^0[\oplus]$ or $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$. In particular, this IPP is used to construct the reduction HIP $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$, in Lemma 4.4.

In what follows, we use the uniformity Definitions 3.15 and 3.16 for triplet languages. In order to do so, we interpret the second and third input for a triplet language as the second input for a pair language.

Theorem 3.26 (Constant-round HIPP for $\mathsf{AC}^0[\oplus]$, $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ [RVW13])**.** *Let $\delta \in (0,1]$ be a constant, and let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) extension fields of $\mathbb{GF}[2]$, where $|\mathbb{H}_n| = \Theta(n^\delta)$ and $|\mathbb{F}_n| = \mathsf{poly}(n)$. Let $\mathcal{L}$ be a triplet language where triplets (x,y,z) have lengths $n_{input} = |x|+|y|+|z| = n_{exp}+n_{imp}+n_{hol} \leq \mathsf{poly}(n)$. Suppose that $\mathcal{L}$ is computable:*

- *either in $s(n)$-succinct and $d(n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$, with circuit size $O(n^C)$, circuit depth $D = O(1)$, succinct description length $s(n) \leq n$, and multiplication gate fan-in $f_{in} = O(n^\delta)$,*
- *or in $s(n)$-succinct and $d(n)$-highly uniform $\mathsf{AC}^0[\oplus]$, with circuit size $O(n^C)$, circuit depth $D = O(1)$, and succinct description length $s(n) \leq n$.*

Then, for every $\varepsilon = \varepsilon(n_{imp}) > 0$ there exists an oblivious holographic $\varepsilon(n)$-IPP for $\mathcal{L}$ with respect to the low degree extension $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ with the following complexity measures:

- *the query complexity to the implicit input is $q = (\frac{1}{\varepsilon(n_{imp})})^{1+o(1)}$,*
- *the number of rounds is $r = O(D \cdot C/\delta)$,*
- *the communication complexity is $\mathsf{cc} = \left((\varepsilon(n_{imp}) \cdot n_{imp}) \cdot n^{O(\delta)} \cdot s(n)^{1+o(1)}\right)$,*
- *the verifier, which gets the $s(n)$-bit string describing the circuit as an additional explicit input, runs in time $n^{O(\delta)} \cdot \left((\varepsilon(n) \cdot n) \cdot s(n)^{1+o(1)} + n_{exp}\right)$,*
- *the honest prover, who gets explicit access to all inputs (including circuit description string) runs in time $\mathsf{poly}(n)$,*
- *the HIPP has perfect completeness and soundness error $\rho = (n^{O(\delta)} \cdot d/|\mathbb{F}|)$.*

As noted above, the IPP is oblivious*: the verifier interacts with the prover without querying the implicit input y (or the holographic input z). At the end of interaction, the verifier either rejects or it outputs a claim about the value of a single coordinate in the low-degree extension of the holographic input z, and a claim about the implicit input y, which is described by:*

1. *an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle Q \rangle$ of the query set $Q \subseteq [n_{imp}]$ as a list of length q (see Definition 3.21). Moreover, the function mapping an index $i \in [q]$ to a query location in $[n_{imp}]$ can be computed by circuits of size $n^{\delta+o(1)}$,*
2. *an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle \psi \rangle$ of the* arithmetic *predicate $\psi : \{0,1\}^q \to \mathbb{F}$ (see Definition 3.20), computed by a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit* of size $(q \cdot n^{\delta+o(1)})$ *over the field $\mathbb{F}$, where $f_{in} = O(\log n)$.*

The HIPP's output claim about the implicit input is that $\psi(y|_Q) = 1$

We stress that the depths of the circuits for Q and ψ are independent of δ. See Appendix A.3 for a review of the [RVW13] construction and the details for the highly-uniform succinct representation of Q and of ψ. As above, the result is shown for $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits. The result for $\mathsf{AC}^0[\oplus]$ circuits then follows using the approximation method (Theorem 3.18).

4 The Protocol for $\mathcal{L}_{\mathsf{HHR}}$

We define the language $\mathcal{L}_{\mathsf{HHR}}$ (stands for *Holographic Hash Root*), with respect to the definition of a hash root from Construction 3.3.

Definition 4.1 (HHR)**.** *The $\mathcal{L}_{\mathsf{HHR}}$ language is parameterized by an ensemble $(\mathbb{F},\mathbb{H})_n$, that defines the low degree extension encoding $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$, by the integers $(n_{in}, n_{out}, n_h, M)_n$, and by a family of functions $\mathcal{H} : \{0,1\}^{n_{in}} \to \{0,1\}^{n_{out}}$. Take $\ell+1$ to be the number of layers in the Merkle Tree (see Construction 3.3) whose leaves are $\mathsf{LDE}(w)$,[14] with respect to $\mathcal{H}$.*

The explicit input to the language is $h = h_1 \circ ... \circ h_\ell$, the concatenation of the length-n_h descriptions of ℓ functions chosen from $\mathcal{H}$, and a string $y \in \{0,1\}^{n_{out}}$. The holographic input is a string $w \in \{0,1\}^M$.

YES *instances of the language are all triplets $\big(w,(y,h)\big)$ such that y is the correct hash root of $\mathsf{LDE}(w)$ with respect to h.*

The following section is devoted to constructing a HIP for $\mathcal{L}_{\mathsf{HHR}}$, whose properties are specified in the theorem below. For soundness, we emphasize that, as in any HIP, the guarantee holds only if the entire string w is *fixed before the protocol begins*, although the verifier never accesses it.

Theorem 4.2. *Let $\delta \in (0,1]$ be a constant, and let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) extension fields of $\mathbb{GF}[2]$, where $|\mathbb{H}_n| = \Theta(n^\delta)$ and $|\mathbb{F}_n| = \mathsf{poly}(n)$. Fix a low degree extension encoding $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$. Set $(n_{in}, n_{out}, n_h, M)_n = (n^{2\delta}, n^\delta, n^{O(\delta)}, \mathsf{poly}(n))$.*

The language $\mathcal{L}_{\mathsf{HHR}}$ as defined in Definition 4.1 has a HIP $(\mathcal{P}_{\mathsf{HHR}}, \mathcal{V}_{\mathsf{HHR}})$ between a prover and a verifier, where the explicit input is $y \in \{0,1\}^{n_{out}}$ and $h \in \{0,1\}^{\ell \cdot n_h}$, and the prover also gets the holographic input $w \in \{0,1\}^M$. The output of the protocol is a coordinate $r \in \mathbb{F}^m$ and a value $v \in \mathbb{F}$ such that:

- **Completeness.** *If $\big(w,(y,h)\big) \in \mathcal{L}_{\mathsf{HHR}}$ and the prover honestly follows the protocol, then $\mathsf{LDE}(w)[r] = v$.*
- **Soundness.** *If $\big(w,(y,h)\big) \notin \mathcal{L}_{\mathsf{HHR}}$, then for any (unbounded) cheating prover, with probability at least $(1 - n^{O(\delta)}/|\mathbb{F}|)$ over the verifier's coins, $\mathsf{LDE}(w)[r] \neq v$.*
- **Complexity.** *The verifier runs in time $n^{O(\delta)}$, the (honest) prover runs in time $\mathsf{poly}(n)$, the number of rounds is $O(1/\delta^3)$ and the communication complexity is $n^{O(\delta)}$.*

4.1 Parameter Setting

For any $\delta \in (0,1]$ and $n \in \mathbb{N}$, we take $\mathbb{H} \subseteq \mathbb{F}$ to be an extension field of $\mathbb{GF}[2]$, such that $|\mathbb{H}| = \Theta(n^\delta)$ and $|\mathbb{F}| = \mathsf{poly}(n)$.[15] Define

$$n_{in} = n^{2\delta},\ n_{out} = n^\delta,$$

[14] We refer to $\mathsf{LDE}(w)$ as the leaves of the tree, although it is not a binary string. See Remark 3.7 for details.

[15] In fact, $|\mathbb{F}|$ can be taken to be a very small polynomial, not much bigger than $|\mathbb{H}|$ (see Section A.2.1). Notice that while $\mathbb{H}$ is a subset of $\mathbb{F}$, and they both extend $\mathbb{GF}[2]$, $\mathbb{H}$ is not necessarily a subfield of $\mathbb{F}$.

and a security parameter $\kappa = n^{\delta}$.[16] Followed by Definition 3.1, we take k to be the smallest integer such that $n_{1_k} = n_{in}$ and $n_{0_k} = n_{out}$. We use a UOWHF family that is secure against polynomial time adversaries (i.e., $\mathsf{poly}(\kappa) = \mathsf{poly}(n)$), such that the time it takes to compute each function is $\mathsf{poly}(\kappa) = n^{O(\delta)}$. Note that if OWFs exist, then there exist UOWHFs satisfying these properties.

Once the fields are set, the length of $|\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(w)| = |\mathsf{LDE}(w)|$ for $|w| = M$ is set, and we can find ℓ: It is the unique solution to the equation

$$\left(\frac{n_{out}}{n_{in}}\right)^{\ell-1} \cdot |\mathsf{LDE}(w)| = n_{out},$$

and this implies that $(n^{\delta})^{\ell} = |\mathsf{LDE}(w)|$, thus $\ell = \log_{n^{\delta}}(|\mathsf{LDE}(w)|)$. We get that $\ell = O(1/\delta)$ for all $M = \mathsf{poly}(n)$, as $|\mathsf{LDE}(w)| = \mathsf{poly}(M)$. In fact, due to Remark 3.7, the length of the leaves is actually $(\log(|\mathbb{F}|) \cdot \mathsf{poly}(M))$, but this is still $\mathsf{poly}(n)$.

4.2 The Protocol

The protocol uses an interactive protocol $(\mathcal{P}_i, \mathcal{V}_i)$ that is implemented in Section 4.3.

The $\mathcal{L}_{\mathsf{HHR}}$ Protocol $(\mathcal{P}_{\mathsf{HHR}}, \mathcal{V}_{\mathsf{HHR}})$

Explicit Input: $y \in \{0,1\}^{n_{out}}, h \in \{0,1\}^{\ell \cdot n_h}$.
Holographic Input: $w \in \{0,1\}^{M}$.

1. The prover $\mathcal{P}_{\mathsf{HHR}}$ creates the Merkle tree with $\mathsf{LDE}(w)$ as leaves and h as the functions, as in Construction 3.3. The tree has $\ell + 1$ layers. Each layer is denoted w_i, where $w_1 = y$ and $w_{\ell+1} = \mathsf{LDE}(w)$. The LDE of each layer is denoted $\hat{w}_i$.
2. The verifier $\mathcal{V}_{\mathsf{HHR}}$ samples an index $r_1 \in_R \mathbb{F}^m$ and sends it to $\mathcal{P}_{\mathsf{HHR}}$. Then, it sets $v_1 = \hat{w}_1[r_1] = \mathsf{LDE}(y)[r_1]$.
3. For $i = 1, ..., \ell$:
   (a) $\mathcal{P}_{\mathsf{HHR}}$ and $\mathcal{V}_{\mathsf{HHR}}$ run $(\mathcal{P}_i, \mathcal{V}_i)$ on explicit input (r_i, v_i) and holographic input w_{i+1}.
   (b) If $\mathcal{V}_i$ rejects then $\mathcal{V}_{\mathsf{HHR}}$ immediately rejects. Otherwise, if $\mathcal{V}_i$ outputs (r_{i+1}, v_{i+1}), then $\mathcal{P}_{\mathsf{HHR}}$ and $\mathcal{V}_{\mathsf{HHR}}$ use this claim for the next iteration.

Output: a claim $(r, v) = (r_{\ell+1}, v_{\ell+1})$ about $\mathsf{LDE}(w) = w_{\ell+1}$.[a]

[a] The claim $(r_{\ell+1}, v_{\ell+1})$ is indeed about $w_{\ell+1}$, rather than about $\hat{w}_{\ell+1}$ (see Remark A.2).

4.3 Constructing the Layer-to-Layer Subprotocol

For the rest of this section we fix some layer $i \in [\ell]$. Our goal is to construct a constant-round HIP $(\mathcal{P}_i, \mathcal{V}_i)$ that reduces the task of verifying that

$$\hat{w}_i[r_i] = v_i,$$

[16] We comment that we could take κ to be even smaller, i.e., n^{ε} for $\varepsilon < \delta$, however taking $\kappa = n^{\delta}$ is sufficient.

to the task of verifying that

$$\hat{w}_{i+1}[r_{i+1}] = v_{i+1},$$

where $r_{i+1} \in \mathbb{F}^m$ (for $m = m(|w_i|) = \lceil \log_{|\mathbb{H}|}(|w_i|) \rceil$) is a random value determined by the verifier, and v_{i+1} is a value determined by the protocol. Since the protocol operates sequentially on blocks of w_i and w_{i+1}, it will be convenient to use the following notation:[17]

$$w_i = y_1, ..., y_k \text{ for } |y_p| = n_{out},$$

and

$$w_{i+1} = z_1, ..., z_k \text{ for } |z_p| = n_{in},$$

for $k = k(i) = |w_{i+1}|/n_{in}$. As noted in Construction 3.3, we assume k is an integer (otherwise, we pad the inputs in a suitable way).

The only exception is the bottom layer of the tree, since unlike any other layer, its extension is also hashed. For this layer, we denote

$$\hat{w}_{\ell+1} = \mathsf{LDE}(w) = z_1, ..., z_k \text{ for } |z_p| = n_{in},$$

i.e., we view its *extension* as divided into blocks, instead of $w_{\ell+1} = w$ itself. The relation between the blocks is, as expected,

$$\forall (i,p) \in [\ell] \times [k] : h_i(z_p) = y_p. \tag{5}$$

For any fixed $i \in [\ell]$, this equation defines a batching relation, as it performs the same computation for k times. Although the blocks of the bottom layer $\hat{w}_{\ell+1}$ are actually over the field $\mathbb{F}$, we think of them as binary strings (we use their binary representation), and thus the whole tree is binary, and the functions are all Boolean. This technical detail is addressed in Remark A.2.

Next, we present the layer-to-layer subprotocol. Its correctness is stated and proved in Lemma 4.6. Before doing so, a few remarks are in order.

First, the protocol uses a sequence of predicates $(\psi_j)_{j \in [\ell']}$. All of them are arithmetic circuits that compute linear combinations of the bits of their inputs, and compare the result to given values. We further discuss their structure in Section 4.4.1, and formally define them in Appendix A.3.

Secondly, we emphasize that when the protocol begins, $\mathcal{P}_i$ has the i^{th} layer $(y_p)_{p \in [k]} = w_i$, as it is invoked by $\mathcal{P}_{\mathsf{HHR}}$ who has full access to w_i. In the first step of the protocol, $\mathcal{P}_i$ computes the tableaux of the computations $(h_i(z_p) = y_p)_{p \in [k]}$, denoted $(\mathcal{T}_p)_{p \in [k]}$. Each y_p will be augmented with the corresponding $\mathcal{T}_p$, and the predicates will operate on subsets of $(y_p, \mathcal{T}_p)_{p \in [k]}$.

Moreover, we take the number of iterations ℓ' to be the maximal integer such that the size of the witnesses that are left before each iteration would be at least $|\mathbb{H}|$ but at most $O(|\mathbb{H}|)$, i.e.,

$$|\mathbb{H}| \leq |y_1 + \mathcal{T}_1| \cdot k^{1-(\ell'-1)\cdot\delta} \leq c \cdot |\mathbb{H}|$$

for some constant $c \geq 1$. In particular, $\ell' \leq 1/\delta + 1$.

We further comment that when $i = \ell$, i.e., when the protocol outputs the claim $(r_{\ell+1}, v_{\ell+1})$ about the bottom layer of the tree, then this claim is about the layer itself, rather than about its low degree extension (or about its binary representation), as it is already given as a low degree extension (see Remark A.2).

[17] Note that $y_1, ..., y_k$ and $z_1, ..., z_k$ depend on the layer $i \in [\ell]$. For the sake of simplicity, this dependence is not captured in our notations.

The layer-to-layer subprotocol $(\mathcal{P}_i, \mathcal{V}_i)$

Explicit Input: $h_i \in \{0,1\}^{n_h}, r_i \in \mathbb{F}^m, v_i \in \mathbb{F}$.
Holographic Input: $(z_p)_{p\in[k]} \in (\{0,1\}^{n_{in}})^k$.

1. $\mathcal{P}_i$ computes $(\mathcal{T}_p)_{p\in[k]}$, the tableaux of the computations $(h_i(z_p) = y_p)_{p\in[k]}$.

2. Both parties set $\langle S_1\rangle$ to be an $n^{\delta+o(1)}$-succinct description (with an empty advice) of the set $S_1 = [k]$ as a list of length k, and $\langle\psi_1\rangle$ to be an $n^{\delta+o(1)}$-succinct description of the arithmetic circuit that is defined as $\psi_1\left((y_p, \mathcal{T}_p)_{p\in S_1}\right) = 1 \iff \mathsf{LDE}(w_i)[r_i] = v_i$.

3. For $j = 1, ..., \ell' - 1$:

   (a) Run $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ on explicit input $(h_i, \langle S_j\rangle, \langle\psi_j\rangle)$, holographic input $(z_p)_{p\in S_j}$ and with respect to parameter $q_j = s_j \cdot k^{-\delta}$, where s_j is the size of the set S_j.[a] More specifically, $\mathcal{V}_i$ emulates $\mathcal{V}_{reduction}$ and $\mathcal{P}_i$ emulates $\mathcal{P}_{reduction}$, and maintains full access to $(y_p, \mathcal{T}_p)_{p\in S_j}$.

   (b) If $\mathcal{V}_{reduction}$ rejects then $\mathcal{V}_i$ immediately rejects. Otherwise $\mathcal{V}_{reduction}$ outputs $\langle S_{j+1}\rangle$ and $\langle\psi_{j+1}\rangle$, as well as a claim (χ_j, θ_j) about $(z_p)_{p\in S_j}$.

4. $\mathcal{P}_i$ sends to $\mathcal{V}_i$ the pairs $(y_p, \mathcal{T}_p)_{p\in S_{\ell'}}$, while $\mathcal{V}_i$ expands $\langle S_{\ell'}\rangle$ to a full description of the list that represents the set $S_{\ell'}$, and $\langle\psi_{\ell'}\rangle$ to a full description of the function $\psi_{\ell'}$. Then, $\mathcal{V}_i$ runs two tests:

   (a) Checks that $\psi_{\ell'}\left((y_p \circ \mathcal{T}_p)_{p\in S_{\ell'}}\right) = 1$.

   (b) $\mathcal{P}_i$ and $\mathcal{V}_i$ run the constant-round HIP of Theorem 3.25 on holographic input $(z_p)_{p\in S_{\ell'}}$ and explicit input $(h_i, (\tilde{y}_p, \tilde{\mathcal{T}}_p)_{p\in S_{\ell'}})$ to check that $\forall p \in S_{\ell'}$, $\tilde{\mathcal{T}}_p$ is the correct tableau for the computation $h_i(z_p) = \tilde{y}_p$.[b] The protocol ends with a claim $(\chi_{\ell'}, \theta_{\ell'})$ about $(z_p)_{p\in S_{\ell'}}$.

5. $\mathcal{P}_i$ and $\mathcal{V}_i$ run the constant-round HIP of Theorem 3.25 on holographic input $(z_p)_{p\in[k]}$ and with respect to the advice strings $(\langle S_j\rangle)_{j\in[\ell']}$ to verify the set of claims $(\chi_j, \theta_j)_{j\in[\ell']}$.[c] The protocol ends with a claim about $(z_p)_{p\in[k]}$, denoted (r_{i+1}, v_{i+1}), and the protocol outputs it.

[a] Technically, it is *the length of the list* that represents the set S_j. It holds that $s_j = q_{j-1}$, where $q_0 = k$.
[b] We denote this relation as $R_{\ell'}$ and define it in Section 4.4.1.
[c] The claims are also considered as a part of the advice for the circuit on which the HIP runs. The full specification of the circuit is defined in Section 4.4.1.

Remark 4.3. *As already done in the protocol statement, throughout this section we abuse notation and do not distinguish between the set S_j and the list that represents it. That is, $(z_p)_{p\in S_j}$ indicates the sequence of holographic inputs according to the order of the* list *that represents the set S_j, rather than according to the set S_j itself, that has no order and no multiplicities. The same is true with respect to $(y_p, \mathcal{T}_p)_{p\in S_j}$.*

We start with an overview of the protocol.

4.3.1 Protocol Overview

We fix $i \in [\ell]$ and restate the goal of the layer-to-layer subprotocol (as stated at the beginning of the section) in terms of $(y_p)_{p\in[k]}$ and $(z_p)_{p\in[k]}$. The protocol reduces two computations:

1. an input claim (r_i, v_i) about $\hat{w}_i[r_i]$, that is, a claim about the low degree extension of $(y_p)_{p\in[k]}$;
2. the relation described in Equation (5), with respect to $(y_p)_{p\in[k]}$ and $(z_p)_{p\in[k]}$,

to an output claim (r_{i+1}, v_{i+1}) about the low degree extension of $(z_p)_{p\in[k]}$, that is, a claim about $\hat{w}_{i+1}$. Since the protocol is *holographic*, its input is divided into an explicit input and holographic input. The claim (r_i, v_i), together with the description of the function h_i, is the explicit input to the protocol. The sequence $(z_p)_{p\in[k]}$ is the holographic input; and, indeed, the protocol outputs a claim about the low degree extension of $(z_p)_{p\in[k]}$. We comment that the sequence $(y_p, \mathcal{T}_p)_{p\in[k]}$ is not an input to the protocol; rather, it is an auxiliary information that the prover has full access to and may use. The verifier does not have access to $(y_p, \mathcal{T}_p)_{p\in[k]}$ and never uses it.

The layer-to-layer subprotocol is inspired by the UP batching protocol of [RRR18] (see Theorem 4.1 in that work). Our protocol assumes the existence of a HIP $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$, that runs recursively on a subset $S \subseteq [k]$ of the input-output pairs (z_j, y_j). The first iteration is instantiated with the set $S_1 = [k]$ and with a predicate ψ_1 such that $\psi_1\left((y_p, \mathcal{T}_p)_{p\in S_1}\right) = 1 \iff \mathsf{LDE}(w_i)[r_i] = v_i$, and they form the explicit input to the reduction protocol. The holographic input is $(z_p)_{p\in S_1}$. When the reduction protocol ends, either the verifier rejects or it outputs a subset $S_2 \subset S_1$ and a predicate ψ_2 computed by the protocol, such that $|S_2| = k^{1-\delta}$ for a suitable constant $\delta > 0$. Moreover, it outputs a claim (χ_1, θ_1) about $(z_p)_{p\in S_1}$. The protocol guarantees that if $\psi_1\left((y_p, \mathcal{T}_p)_{p\in S_1}\right) = 0$, then w.h.p. either $\psi_2\left((y_p, \mathcal{T}_p)_{p\in S_2}\right) = 0$ or (χ_1, θ_1) is a false claim (i.e., that $\mathsf{LDE}\left((z_p)_{p\in S_1}\right)[\chi_1] \neq \theta_1$).

The claim (χ_1, θ_1) is kept aside until the end of all iterations. Notice that at this point, we are in better shape than where we started, because the computation that needs to be verified refers to a *smaller* (holographic) input. (S_2, ψ_2) are the explicit input to the second iteration, and the holographic input is narrowed down to $(z_p)_{p\in S_2}$. When it ends, once again if the verifier does not reject, then it outputs a subset $S_3 \subset S_2$, a predicate ψ_3 and a claim (χ_2, θ_2) about $(z_p)_{p\in S_2}$, such that $|S_3| = k^{1-2\delta}$, and the soundness guarantee is defined analogously.

After $(\ell' - 1) = O(1/\delta)$ iterations, if the verifier does not reject, then we are left with a set $S_{\ell'}$ that contains only a few elements out of the holographic input, and with a predicate $\psi_{\ell'}$. With high probability, it holds that $\psi_1\left((y_p, \mathcal{T}_p)_{p\in S_1}\right) = 1$ if and only if $\psi_{\ell'}\left((y_p, \mathcal{T}_p)_{p\in S_{\ell'}}\right) = 1$ and $(\chi_j, \theta_j)_{j\in[\ell'-1]}$ are all correct claims about the holographic input.

The verifier cannot check the result of $\psi_{\ell'}$ by itself, although $S_{\ell'}$ is small, since it does not have any access to $(y_p, \mathcal{T}_p)_{p\in S_{\ell'}}$. Nor it can check the consistency between $(y_p, \mathcal{T}_p)_{p\in S_{\ell'}}$ and $(z_p)_{p\in S_{\ell'}}$ (namely, that $h_i(z_p) = \tilde{y}_p$ for every $p \in S_{\ell'}$ and that $\tilde{\mathcal{T}}_p$ is the correct tableau for this computation), because it does not have explicit access to $(z_p)_{p\in S_{\ell'}}$, rather, only *holographic* access. A natural solution is asking from the prover to send over $(y_p, \mathcal{T}_p)_{p\in S_{\ell'}}$. After that, the verifier can check that the alleged $(\tilde{y}_p, \tilde{\mathcal{T}}_p)_{p\in S_{\ell'}}$ satisfy $\psi_{\ell'}$, and that they are consistent with the holographic input $(z_p)_{p\in[k]}$. The latter check is performed by executing another holographic IP, that (interactively) reduces this task to a single claim about the holographic input, that the protocol outputs. In fact, the task of verifying the set of claims $(\chi_j, \theta_j)_{j\in[\ell'-1]}$ that we kept so far is also taken into account in this reduction.

To conclude the overview, we demistify one last technical detail, regarding the representation of the sets. Both the implicit and the holographic inputs in each iteration are ordered according

to *a list that represents the set* S_j (see Definition 3.21). We note that the reduction protocol does not output (descriptions of) ordered lists. Rather, each $S_j \subseteq [k]$ is represented by a list of indices in an arbitrary order (determined by the protocol, or in particular, by the verifier's coin tosses), and moreover, this list may have multiplicities. Sorting this list and eliminating multiplicities indeed gives the set S_j, but the verifier cannot do this by itself, as this list may be as long as k. Nevertheless, the succinctness condition that all of these sets satisfy promises that the verifier *is* able to delegate the task of "sorting" these lists, and indeed this is also taken into account in the holographic IP that we run in the last step.

4.4 Proving the Layer-to-Layer Subprotocol

Roadmap for Section 4.4. Theorem 3.26 presented the HIPP used in the layer-to-layer subprotocol, that is applicable to any sufficiently uniform triplet language. We use this HIPP to instantiated Lemma 4.4, that implements the HIP $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$. After we prove the correctness of the reduction subprotocol, we use it in Lemma 4.6 to prove the correctness of the layer-to-layer subprotocol $(\mathcal{P}_i, \mathcal{V}_i)$. Finally, in Section 4.4.1, we prove that the relations on which we applied the HIPP indeed satisfy the required uniformity conditions.

First, let us formally define the "batching" relation R_S, which is the relation that the HIPP built in Theorem 3.26 is going to work on. For that, we first formally define R, the batching relation of Equation (5). Since we do not want to assume anything about h_i, and, in particular, that it is computable in AC^0, we augment the relation with the tableau of h_i's computation. This is captured by the relation R:

$$\big((h_i, z), (y, \mathcal{T})\big) \in R \iff h_i(z) = y \text{ and } \mathcal{T} \text{ is the correct tableau for computing } h_i(z) = y. \quad (6)$$

Loosely speaking, although $R \in \mathsf{P}$, it can be viewed as a UP-relation that checks the validity of a hash value y, given a string z and a function h_i. That is, y is considered the correct witness for (h_i, z) if and only if $h_i(z) = y$. As already mentioned, we augment the witness y with the tableau $\mathcal{T}$ of the computation $h_i(z) = y$, in order to reduce the complexity of the verification to AC^0. We stress that the tableau is *unique*, thus every (h_i, z) has only one unique witness. The complexity and uniformity conditions that R has to satisfy are proved in Claim 4.7 in Section 4.4.1.

Next, we turn to define R_S. We generalize the definition of an NP relation for inputs of the form (x, z), and witnesses y. We think of x as some additional information that may be common to many different pairs (z, y). Taking x to be empty gives a standard NP relation.

Fix two descriptions $\langle S\rangle, \langle\psi\rangle$ of a set S and a predicate ψ as described above. For a (generalized) NP relation R, we consider a related (generalized) NP relation R_S, viewed as a triplet language and defined as:

$$R_S \stackrel{\text{def}}{=} \Big\{ \big(\overbrace{x, \langle S\rangle, \langle\psi\rangle}^{\text{explicit}}, \overbrace{(y_p)_{p\in S}}^{\text{implicit}}, \overbrace{(z_p)_{p\in S}}^{\text{holographic}}\big) : \forall p \in S, \big((x, z_p), y_p\big) \in R \text{ and } \psi((y_p)_{p\in S}) = 1 \Big\}. \quad (7)$$

We defer the uniformity proof of R_S to Section 4.4.1 as well (see Claim 4.8 there).

The following lemma is a generalization of Lemma 4.1.1 in [RRR18], in the sense that it uses a HIPP instead of an IWV, and builds a HIP instead of an IP. The lemma corresponds to a single step in reducing the number of statements. Loosely speaking, this step shows an interactive protocol, where if we start with a false claim about a subset of the k UP statements, then at the end of the

protocol, with high probability, we will have a false claim about a smaller subset of the statements. This step is where we rely on the existence of the oblivious HIPP for sufficiently uniform AC^0 triplet language. For an NP relation with input of the form (x, z) and witnesses y, we use the HIPP in the natural way: x is the explicit input, z is the holographic input and y is the implicit input. As mentioned earlier, Lemma 4.4 can be instantiated with any such HIPP. The HIP is derived almost directly from the HIPP, and this lemma takes care of the technical details that arise when using the HIPP for constructing a batching HIP.

Lemma 4.4. *Let $\delta \in (0, 1]$ be a constant, and let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) extension fields of $\mathbb{GF}[2]$, where $|\mathbb{H}_n| = \Theta(n^\delta)$ and $|\mathbb{F}_n| = \mathsf{poly}(n)$.*

Let $\mathcal{L}$ be a triplet language with input of length $n_{input} = n_{exp}+n_{imp}+n_{hol}$, where $n_{input} \leq \mathsf{poly}(n)$. Suppose that if $\mathcal{L}$ is computable by an ensemble of $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ or $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits with $f_{in} = O(n^\delta)$, then for every parameter q, $\mathcal{L}$ has an oblivious HIPP with respect to the low degree extension $\mathsf{LDE} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ with the following complexity measures:

- *the proximity parameter is $\varepsilon(n_{input}) = n^\delta_{imp}/q$,*
- *the number of rounds is $r = r(n_{exp}, n_{imp}, n_{hol}, \delta)$,*
- *the communication complexity is $\mathsf{cc} = \mathsf{cc}(n_{exp}, n_{imp}, n_{hol}, q)$,*
- *the verifier, which gets the $n^{\delta+o(1)}$-bit string describing the circuit as an additional explicit input, runs in time $\mathcal{V}\mathsf{time} = \mathcal{V}\mathsf{time}(n_{exp}, n_{imp}, n_{hol}, q)$,*
- *the honest prover, who gets explicit access to all inputs (including circuit description string) runs in time $\mathcal{P}\mathsf{time} = \mathcal{P}\mathsf{time}(n_{exp}, n_{imp}, n_{hol}, q)$,*
- *the HIPP has perfect completeness and soundness error $\rho = \rho(n_{exp}, n_{imp}, n_{hol}, q)$,*

such that the query set $Q \subseteq [n_{imp}]$ that the HIPP generates has an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle Q\rangle$ as a list of length q, where the function mapping an index $i \in [q]$ to a query location in $[n_{imp}]$ is computed by circuits of size $n^{\delta+o(1)}$; and, the predicate $\psi : \{0, 1\}^q \to \mathbb{F}$, which represents the claim about the implicit input, has an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle\psi\rangle$ and is computed by a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size of size $(q\cdot n^{\delta+o(1)})$, where $f_{in} = O(n^\delta)$.

Let R be a UP relation computable in n^δ-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$, with inputs of the form (x, z) and witnesses y. There exists a HIP between a prover $\mathcal{P}_{reduction}$ and a verifier $\mathcal{V}_{reduction}$ with respect to the same low degree extension LDE, such that the following holds. The explicit input is x and $n^{\delta+o(1)}$-succinct descriptions $\langle S\rangle$ that satisfies the uniformity condition described above, and $\langle\psi\rangle$ that describes a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit, where $f_{in} = O(n^\delta)$. The holographic input is $z_S = (z_p)_{p\in S}$. In addition, the prover $\mathcal{P}_{reduction}$ also gets access to the witnesses $y_S = (y_p)_{p\in S}$. The two parties interact and at the end of the interaction $\mathcal{V}_{reduction}$ either rejects or it outputs:

1. *an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle S'\rangle$ of a set $S' \subseteq S$ as a list of length q, where the function mapping an index $i \in [q]$ to a query location in S is computed by circuits of size $n^{\delta+o(1)}$;*
2. *an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle\psi'\rangle$ of the predicate $\psi : \{0, 1\}^{q\cdot n_{imp}} \to \mathbb{F}$, computed by a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size $(q \cdot n^{\delta+o(1)} + \tilde{O}(n_{imp}))$, where $f_{in} = O(n^\delta)$;*

3. *a claim* (χ, θ) *about* $\mathsf{LDE}(z_S) = \hat{z}_S$*;*

such that:

- **Completeness.** *If* $(x, z_p, y_p) \in R$ *for all* $p \in S$*, and* $\psi(y_S) = 1$*, then, with probability 1, after interacting with* $\mathcal{P}_{reduction}$*, the verifier* $\mathcal{V}_{reduction}$ *outputs* $\langle S' \rangle, \langle \psi' \rangle, (\chi, \theta)$ *such that*

$$\psi'(y_{S'}) = 1,$$

and

$$\hat{z}_S[\chi] = \theta.$$

- **Soundness.** *If either (1) there exists* $p \in S$ *such that* $R(x, z_p) = \emptyset$*, or (2)* $(x, z_p, y_p) \in R$ *for all* $p \in S$ *but* $\psi(y_S) = 0$*, then, for every prover strategy* $\mathcal{P}^*_{reduction}$*, with probability* $1 - \rho$*, after interacting with* $\mathcal{P}^*_{reduction}$*, the verifier* $\mathcal{V}_{reduction}$ *either rejects or outputs* $\langle S' \rangle, \langle \psi' \rangle, (\chi, \theta)$ *such that one of the following holds:*

  1. $\exists p \in S'$ *such that* $R(x, z_p) = \emptyset$*; or*
  2. $\psi'(y_{S'}) = 0$*; or*
  3. $\hat{z}_S[\chi] \neq \theta$*.*

- **Complexity.** *Taking* $n_{exp} = |x| + |\langle S \rangle| + |\langle \psi \rangle| + |\xi|$*,* $n_{imp} = s \cdot |y_1|$ *and* $n_{hol} = s \cdot |z_1|$ *and* $\varepsilon = n_{imp}^{\delta}/q$*, the protocol holds the same complexities of the HIPP.*

Proof. Fix $\delta, \mathbb{H}, \mathbb{F}$ as stated above. Let R be a UP relation computable in n^{δ}-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$, with inputs of the form (x, z) and witnesses y.

Let R_S be as defined in Equation (7). By Claim 4.8, R_S is computable by an ensemble of $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits with $f_{in} = O(n^{\delta})$. Therefore, by the lemma's hypothesis, there exists an oblivious HIPP $(\mathcal{P}_{IPP}, \mathcal{V}_{IPP})$ for R_S, where we set the proximity parameter to be $\varepsilon = n_{imp}^{\delta}/q$.

As mentioned in Remark 3.19, the first message of the HIPP is a random seed of length $\mathsf{polylog}(n)$ that the verifier uniformly selects. Then, both parties follow the protocol, and at the end the prover sends an additional message. This change does not increase the communication complexity of the HIPP, nor hurts its soundness or the parties' running time, but we still include its length as a part of the explicit input to R_S, on which we run the HIPP. However, for simplicity, we omit it when we consider the input to R_S.

We use $(\mathcal{P}_{IPP}, \mathcal{V}_{IPP})$ to construct a protocol $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ as required in the theorem's statement. The verifier $\mathcal{V}_{reduction}$ and the prover $\mathcal{P}_{reduction}$ run $(\mathcal{P}_{IPP}, \mathcal{V}_{IPP})$ with respect to the explicit input $(x, \langle S \rangle, \langle \psi \rangle)$, the holographic input $z_S = (z_p)_{p \in S}$ and the implicit input $y_S = (y_p)_{p \in S}$ (recall that $\mathcal{P}_{reduction}$ has full access the witnesses).

If $\mathcal{V}_{IPP}$ rejects then $\mathcal{V}_{reduction}$ immediately rejects. Otherwise, $\mathcal{V}_{IPP}$ outputs the descriptions $\langle Q' \rangle$ and $\langle \psi'_{int} \rangle$ where $Q' \subseteq S \times [n_{imp}]$, of size q, specifies which bits to read from each of the witnesses included in y_S, while $\psi'_{int} : \{0,1\}^q \to \{0,1\}$ is a predicate specifying whether $\mathcal{V}_{IPP}$ would have accepted had it read those bits. Let $S' \subset S$ denote the witnesses that Q' refers to, and let $\psi' : \{0,1\}^{q \cdot n_{imp}} \to \{0,1\}$ be the predicate that gets the q chosen witnesses entirely and returns the same answer that ψ'_{int} returns on their appropriate subset of size q. Moreover, $\mathcal{V}_{IPP}$ outputs a claim (χ, θ) about $\hat{z}_S$. Overall, the verifier $\mathcal{V}_{reduction}$ outputs $\langle S' \rangle$, $\langle \psi' \rangle$ and the claim (χ, θ).

Succinctness of S'. Without loss of generality, we assume that k and n_{imp} are powers of 2. This means that for every coordinate $r \in \{0,1\}^{\log k + \log(n_{imp})}$ (the representation of a coordinate in $Q' \subseteq [k] \times [n_{imp}]$), the first $\log k$ bits indicate which witness r refers to, and the other bits indicate the locations to read from the witness in this index.

Since Q' has an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle Q' \rangle$ as a list of length q, there exists an $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit $C_{Q'}$ that maps indices in $[q]$ to elements in $S \times [n_{imp}]$, and fully describes Q' (see Definition 3.21). By the assumption, $C_{Q'}$ is of size $n^{\delta+o(1)}$

We describe a circuit C_{chop} that works as follows: on input $r \in \{0,1\}^{\log k + \log(n_{imp})}$, it returns the first $\log k$ bits of r. It is immediate that C_{chop} is an $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit, and of size smaller than $n^{\delta+o(1)}$.

The circuit $C_{S'}$ composes $C_{Q'}$ and C_{chop}, by relabeling the input gates for C_{chop} to be the output gates of $C_{Q'}$, and using $\langle Q' \rangle$ as its advice string (i.e., $\langle S' \rangle = \langle Q' \rangle$). Following the same arguments from the proof of Claim 4.8, this implies that the circuit $C_{S'}$ is an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit of size $n^{\delta+o(1)}$, that maps indices in $[q]$ to elements in S, such that the definition for succinct sets is satisfied. We conclude that S' has an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description as a list of length q.

Uniformity of ψ'. Recall that $\psi'_{int} : \{0,1\}^q \to \{0,1\}$ has an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description $\langle \psi'_{int} \rangle$ and is computed by a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size $(q \cdot n^{\delta+o(1)})$, where $f_{in} = O(n^\delta)$. Take $C_{\psi'_{int}}$ to be that circuit.

The circuit $C_{\psi'}$ uses as advice the concatenation of the advice of $C_{\psi'_{int}}$ and the advice of Q' (i.e., $\langle \psi' \rangle = \langle \psi'_{int} \rangle \circ \langle Q' \rangle$). First, it uses $\langle Q' \rangle$ in order to construct the circuit $C_{Q'}$[18] as described above. In runs q copies of it in parallel, where it feeds the j^{th} copy with the constant j. Recall that the witnesses that are the input to $C_{\psi'}$ are ordered by S', therefore if the output of the circuit $C_{Q'}$ on j is $(j,i) \in S \times [n_{imp}]$, then $y_j[i]$ is exactly the j^{th} input of $C_{\psi'_{int}}$ (where these indices are given in binary). In order to feed $C_{\psi'_{int}}$ with the i^{th} bit of the j^{th} witness, we use the circuit C_{select} described next. Although this circuit is arithmetic, it only works on Boolean values, and this fact is crucial in its construction.

Given a string y and an index $i \in \{0,1\}^{\log |y|} = \{0,1\}^{\log(n_{imp})}$, the circuit C_{select} returns y_i. This circuit C_{select} is already built in Section 4.4.1, and we use it with $|z_1| = 1$ and $k = n_{imp}$. The circuit C_{select} is an $O(\log(n_{imp}))$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size $\tilde{O}(n_{imp})$, where $f_{in} = \log(n_{imp})$. Since $n_{input} \leq \mathsf{poly}(n)$, in particular $\log(n_{imp}) = O(\log n)$.

Next, $C_{\psi'}$ uses $\langle \psi'_{int} \rangle$ in order to construct the circuit $C_{\psi'_{int}}$. As mentioned earlier, it is $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size $(q \cdot n^{\delta+o(1)})$, where $f_{in} = O(n^\delta)$. The circuit $C_{\psi'}$ connects the j^{th} input of $C_{\psi'_{int}}$ to be the output of $C_{\mathsf{select}}(y_j, i)$. The relabeling of the gates and the composition of the adjacency predicates and incidence functions is done in the natural way, as described in Claim 4.8.

Since $|\langle \psi' \rangle| = |\langle \psi'_{int} \rangle| + |\langle Q' \rangle| = 2n^{\delta+o(1)} = n^{\delta+o(1)}$, we conclude that $\psi : \{0,1\}^{q \cdot n_{imp}} \to \mathbb{F}$ has an $n^{\delta+o(1)}$-succinct description $\langle \psi' \rangle$ and that it is computed by an $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit of size $(q \cdot n^{\delta+o(1)} + \tilde{O}(n_{imp}))$, where $f_{in} = O(n^\delta)$, as stated.

We comment that the depths of $C_{S'}$ and $C_{\psi'}$ are some global constants independent of δ, since by the assumption, $C_{Q'}$ and $C_{\psi'_{int}}$ satisfy this property as well.

[18] By that, we mean that the machine that construct this circuit uses this advice string.

Completeness. Let z_S be a sequence of holographic inputs, and let x be an explicit input together with a set $S \subseteq [k]$ and a circuit ψ. Let be a random seed ξ, and assume that there exists a unique y_S such that $(z_p, y_p) \in R$ for all $p \in S$ and $\psi(y_S) = 1$. The HIPP is run with respect to an input $\big((x, \langle S\rangle, \langle\psi\rangle, z_S), y_S\big) \in R_S$ and ξ. Thus, by the completeness of the HIPP and Remark 3.19, with probability 1, it holds that $\psi'(y_{S'}) = 1$ and that $\hat{z}_S[\chi] = \theta$.

Soundness. Fix a seed ξ. According to Theorem 3.18, with probability at most $2^{-\log^2 n}$ (recall that in Claim 4.8 we took the error function to be $\lambda(n) = \log^2 n$), it holds that ξ is not a good seed for approximating one of the circuits which compose $C_{\wedge R}$. Taking a Union Bound over these $q \leq n$ circuits that we approximate, we get an error probability of at most $n \cdot 2^{-\log^2 n} = o(1)$. Therefore, with probability $1 - o(1)$, the seed is a good choice for the circuit which computes R_S. In the rest of the analysis, we assume that the verifier was not extremely unlucky and that this event does not happen.

Suppose that either there exists $p \in S$ such that $R(x, z_p) = \emptyset$, or $\psi(y_S) = 0$. Let $\mathcal{P}^*_{reduction}$ be a cheating prover strategy. To show that the soundness condition holds, it suffices to prove the following claim:

Claim 4.5.

$$\Pr[(\forall p \in S',\ R(x, z_p) \neq \emptyset)\ \textit{and}\ (\psi(y_{S'}) = 1)\ \textit{and}\ (\hat{z}_S[\chi] = \theta)] \leq \rho.$$

Proof. First, we observe that the assumption implies that $R_S(x, \langle S\rangle, \langle\psi\rangle, z_S) = \emptyset$: If $\exists p \in S$ such that $R(x, z_p) = \emptyset$, then the first condition for membership in R_S is violated, and if $\psi(y_S) = 0$, then the second condition for membership in R_S is violated.

For every $p \in S$, if $R(x, z_p) \neq \emptyset$ then define $\bar{y}_p = y_p$ (i.e., the unique witness for (x, z_p)), whereas if $R(x, z_p) = \emptyset$ then define $\bar{y}_p$ as some arbitrary fixed string (e.g., $0^{n_{imp}}$).

We view $\mathcal{P}^*_{reduction}$ as an adversary for the oblivious HIPP, with respect to the a priori fixed witness string $\bar{y}_S = (\bar{y}_p)_{p \in S}$. By the soundness condition of the HIPP, it holds that:

$$\Pr[\psi'(\bar{y}_{S'}) = 1] \leq \rho.$$

For all $p \in S$, we have that if $R(x, z_p) \neq \emptyset$ then $y_p = \bar{y}_p$. Thus,

$$\begin{aligned}&\Pr[(\forall p \in S',\ R(x, z_p) \neq \emptyset) \text{ and } (\psi'(y_{S'}) = 1) \text{ and } (\hat{z}_S[\chi] = \theta)] \leq \\ &\Pr[(\psi'(\bar{y}_{S'}) = 1) \text{ and } (\hat{z}_S[\chi] = \theta)] \leq \rho,\end{aligned}$$

and the claim follows. □

Complexity. The stated complexity follows from the complexity of the HIPP run on a relation R_S, when setting $\varepsilon = n^{\delta}_{imp}/q$, which yields:

- $n_{exp} = |x| + |\langle S\rangle| + |\langle\psi\rangle| + |\xi|$,
- $n_{imp} = |y_S| = s \cdot |y_1|$,
- $n_{hol} = |z_S| = s \cdot |z_1|$.

□

Finally, using the aforementioned results, we are ready to prove the correctness of the layer-to-layer subprotocol $(\mathcal{P}_i, \mathcal{V}_i)$.

Lemma 4.6. *Let* $\overbrace{h_i, r_i, v_i}^{\text{explicit input}}, \overbrace{(z_p)_{p\in[k]}}^{\text{holographic input}}, \overbrace{(y_p, \mathcal{T}_p)_{p\in[k]}}^{\text{witnesses}}$ *be as described above, and let* $\delta > 0$ *be a constant. The interactive protocol* $(\mathcal{P}_i, \mathcal{V}_i)$ *presented in Protocol 4.3 has the following properties:*

- **Completeness.** *If* $\hat{w}_i[r_i] = v_i$*, then, with probability 1, after interacting with* $\mathcal{P}_i$*, the verifier* $\mathcal{V}_i$ *outputs* (r_{i+1}, v_{i+1}) *such that* $\hat{w}_{i+1}[r_{i+1}] = v_{i+1}$.
- **Soundness.** *If* $\hat{w}_i[r_i] \neq v_i$*, then for every prover strategy* $\mathcal{P}_i^*$*, with probability* $(1 - n^{O(\delta)}/|\mathbb{F}|)$*, after interacting with* $\mathcal{P}_i^*$*, the verifier* $\mathcal{V}_i$ *either rejects or outputs* (r_{i+1}, v_{i+1}) *such that* $\hat{w}_{i+1}[r_{i+1}] \neq v_{i+1}$.
- **Complexity.** *The verifier runs in time* $n^{O(\delta)} \cdot (k^{2\delta} + 1)$*, the (honest) prover runs in time* $\mathsf{poly}(n, k)$*, the number of rounds is* $O(1/\delta^2)$ *and the communication complexity is* $n^{O(\delta)} \cdot (k^{2\delta} + 1)$.

Proof. Let $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ be the protocol guaranteed by Lemma 4.4, with respect to the UP relation R and the HIPP of Theorem 3.26. In detail, in the j^{th} iteration of the protocol, for $j \in [\ell' - 1]$, we use $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ for the relation R_S as defined in Equation (7), with respect to R as defined in Equation (6). We use the parameter $q_j = s_j \cdot k^{-\delta}$, where s_j is the length of the list that represents the set S_j. This means that the length of list that represents the set of remaining witnesses decreases by a factor of k^δ in each iteration. We get that $y_{S_j} = (y_p, \mathcal{T}_p)_{p\in S_j}$ and that the explicit input is h_i (denoted x), together with $\langle S_j \rangle$ and $\langle \psi_j \rangle$.

Notice that both of these usages rely on the fact that the relation R satisfies a uniformity requirement, and indeed this is proved in Claim 4.7.

Completeness. Assume $\hat{w}_i[r_i] = v_i$. Let $(y_p, \mathcal{T}_p)_{p\in[k]}$ be the correct unique witnesses for $(z_p)_{p\in[k]}$, such that for every $p \in [k]$, $\big((h_i, z_p), (y_p, \mathcal{T}_p)\big) \in R$. Let $S_1, ..., S_{\ell'}$ and $\psi_1, ..., \psi_{\ell'}$ be the sets and the formulas, respectively, generated in the interaction between $\mathcal{P}_i$ and $\mathcal{V}_i$.

By the definition of ψ_1, it holds that $\psi_1\left((y_p, \mathcal{T}_p)_{p\in S_1}\right) = \psi_1\left((y_p, \mathcal{T}_p)_{p\in[k]}\right) = 1$. By the perfect completeness of the $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ protocol, for each iteration $j \in [\ell' - 1]$, since $\big((h_i, z_p), (y_p, \mathcal{T}_p)\big) \in R$ for all $p \in S_j$ and $\psi_j\left((y_p, \mathcal{T}_p)_{p\in S_j}\right) = 1$, then $\mathcal{V}_{reduction}$ does not rejects, $\psi_{j+1}\left((y_p, \mathcal{T}_p)_{p\in S_{j+1}}\right) = 1$ and (χ_j, θ_j) is a correct claim about $(z_p)_{p\in S_j}$. Thus, at the end of the loop, $\psi_{\ell'}\left((y_p, \mathcal{T}_p)_{p\in S_{\ell'}}\right) = 1$, and, moreover, all claims $(\chi_j, \theta_j)_{j\in[\ell']}$ are correct claims. Since $\mathcal{P}_i$ sends the correct witnesses $(y_p, \mathcal{T}_p)_{p\in S_{\ell'}}$, by the perfect completeness of the constant-round HIP of Theorem 3.25, the verifier $\mathcal{V}_i$ outputs correct claim $(\chi_{\ell'}, \theta_{\ell'})$ about $(z_p)_{p\in S_{\ell'}}$ in Step (4b). This means that all of the claims $(\chi_j, \theta_j)_{j\in[\ell']}$ in Step (5) are reduced into a single correct claim (r_{i+1}, v_{i+1}) with probability 1, again by the perfect completeness of Theorem 3.25.

Soundness. For the soundness condition, fix $\hat{w}_i, r_i$ and v_i such that $\hat{w}_i[r_i] \neq v_i$, and fix $\hat{w}_{i+1}$. Take A to be the event that after interacting with a cheating prover $\mathcal{P}_i^*$, the verifier $\mathcal{V}_i$ does not reject and outputs a correct claim about $\hat{w}_{i+1}$. We wish to bound

$$\Pr[A] = \Pr[\mathcal{V}_i \text{ does not reject and } \hat{w}_{i+1}[r_{i+1}] = v_{i+1}].$$

Recall that the protocol $(\mathcal{P}_i, \mathcal{V}_i)$ consists of $\ell' - 1$ phases. Each phase consists of an interactive protocol $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ with soundness error ρ. For every $1 \leq j \leq \ell' - 1$, let T_j denote the

event that the verifier does not reject in the j^{th} phase and that its output satisfies the following two condition:

1. $\psi_{j+1}\left((y_p, \mathcal{T}_p)_{p\in S_{j+1}}\right) = 1$;
2. The claim about z_{S_j} is correct, i.e., $\hat{z}_{S_j}[\chi_j] = \theta_j$.

Thus, assuming that $\hat{w}_i[r_i] \neq v_i$ implies that $(\neg T_1)$ (because, in this case, the first condition is violated by the definition of ψ_1). Let E denote the event that at least one of the claims $(\chi_j, \theta_j)_{j\in[\ell']}$ reduced in Step (5) is false, but the output claim (r_{i+1}, v_{i+1}) is correct. Let W denote the event that $\exists p^* \in S_{\ell'}$ s.t. $(\tilde{y}_{p^*}, \tilde{\mathcal{T}}_{p^*}) \neq (y_{p^*}, \mathcal{T}_{p^*})$ (indeed, all of the pairs $(y_p, \mathcal{T}_p)$ are well-defined once $\hat{w}_i$ is fixed). By elementary probability theory,

$$\begin{aligned}\Pr[A] &= \Pr\left[(E \wedge A) \vee \left(\neg E \wedge \left((W \wedge A) \vee (\neg W \wedge A)\right)\right)\right] \\ &\leq \Pr[E \wedge A] + \Pr[\neg E \wedge W \wedge A] + \Pr[\neg E \wedge \neg W \wedge A].\end{aligned}$$

First, Theorem 3.25 implies that $\Pr[E \wedge A] \leq n^{O(\delta)}/|\mathbb{F}|$.

Secondly, we recall that z_{p^*} is *fixed* and cannot be chosen by the prover, as the $i+1^{\text{st}}$ layer is fixed (and $w_{i+1} = z_1 \circ ... \circ z_k$). This means that if $\exists p^* \in S_{\ell'}$ s.t. $(\tilde{y}_{p^*}, \tilde{\mathcal{T}}_{p^*}) \neq (y_{p^*}, \mathcal{T}_{p^*})$, then the input in Step (4b) does not satisfy the relation $R_{\ell'}$ as defined in the protocol (because $h_i(z_{p^*}) \neq \tilde{y}_{p^*}$). However, by assuming $(\neg E)$, we know that all of the claims that are reduced in Step (5) are correct, and in particular, $(\chi_{\ell'}, \theta_{\ell'})$ as well. Thus, the event $(\neg E \wedge W \wedge A)$ implies that the prover breaks the soundness guarantee of the protocol of Theorem 3.25, which in turn cannot happen with probability bigger than $n^{O(\delta)}/|\mathbb{F}|$.

Finally, observe that the event $(\neg E \wedge \neg W \wedge A)$ implies $T_{\ell'}$, since otherwise the verifier rejects in Step (4a) (notice that this is due to the assumption $(\neg W)$, which means that the test in Step (4a) is applied to the correct witnesses $(y_{p^*}, \mathcal{T}_{p^*})$). We already know that $(\neg T_1)$, hence there exists a phase $j \in [\ell' - 1]$ such that $(\neg T_j \wedge T_{j+1})$, i.e., that the soundness guarantee of the HIPP does not hold, which happens with probability ρ. This implies that

$$\Pr[\neg E \wedge \neg W \wedge A] \leq \Pr[\exists j \text{ s.t. } \neg T_j \wedge T_{j+1} \wedge A] \leq \sum_{j=1}^{\ell'-1} \Pr[\neg T_j \wedge T_{j+1} \wedge A] \leq (\ell' - 1) \cdot \rho.$$

All in all, we get that

$$\Pr[A] \leq \frac{n^{O(\delta)}}{|\mathbb{F}|} + \frac{n^{O(\delta)}}{|\mathbb{F}|} + (\ell' - 1) \cdot \rho.$$

Since $\rho = n^{O(\delta)} \cdot O(\log n)/|\mathbb{F}| = n^{O(\delta)}/|\mathbb{F}|$, since $d = O(\log n)$ (all of the circuits are $O(\log n)$-highly uniform), and $\ell' \leq 1/\delta$, we conclude that $\Pr[A] \leq n^{O(\delta)}/|\mathbb{F}|$ as desired.

Complexity. First, we consider the complexity measures of running the reduction subprotocol. For every $j \in [\ell']$, let s_j denote the length of the list that represents the set S_j. Since the descriptions of S_j and ψ_j are $n^{\delta+o(1)}$-succinct, we get that $|\langle\psi_j\rangle| + |\langle S_j\rangle| = n^{\delta+o(1)}$. Moreover, each h_i can be computed in time $n^{O(\delta)}$, which implies that $|x| = n_h = n^{O(\delta)}$ and $|\mathcal{T}_j| = n^{2\cdot O(\delta)} = n^{O(\delta)}$.

Consider the j^{th} iteration of the loop, for some $j \in [\ell' - 1]$. Let n^j_{exp}, n^j_{hol} and n^j_{imp} denote the lengths of the explicit, holographic and implicit inputs to the j^{th} iteration. We get

$$n^j_{exp} = |x| + |\langle S_j\rangle| + |\langle\psi_j\rangle| + |\xi| \leq n^{O(\delta)} + 2n^{\delta+o(1)} + \mathsf{polylog}(n) = n^{O(\delta)}.$$

As for the implicit and holographic inputs,

$$n^j_{imp} = |S_j| \cdot (|y_1| + |\mathcal{T}_1|) = s_j \cdot (n_{out} + n^{O(\delta)}) \leq s_j \cdot n^{O(\delta)},$$

and

$$n^j_{hol} = s_j \cdot |z_1| = s_j \cdot n_{in}.$$

Recall that $q_j = s_j \cdot k^{-\delta}$. By Theorem 3.26, together with Lemma 4.4, the j^{th} iteration takes $O(1/\delta)$ rounds and has:

- Communication complexity:

$$\begin{aligned}\mathsf{cc}_j &= \frac{(n^j_{imp})^\delta}{s_j \cdot k^{-\delta}} \cdot n^j_{imp} \cdot n^{O(\delta)} \cdot (n^{\delta+o(1)})^{1+o(1)} \\ &\leq \frac{s_j^\delta \cdot n^{O(\delta^2)}}{s_j \cdot k^{-\delta}} \cdot s_j \cdot n^{O(\delta)} \cdot n^{O(\delta)} \leq k^{2\delta} \cdot n^{O(\delta)}\end{aligned}$$

- Verifier running time:

$$\begin{aligned}\mathcal{V}\mathsf{time}_j &= n^{O(\delta)} \cdot \left(\frac{(n^j_{imp})^\delta}{s_j \cdot k^{-\delta}} \cdot n^j_{imp} \cdot (n^{\delta+o(1)})^{1+o(1)} + n^j_{exp} \right) \\ &\leq n^{O(\delta)} \cdot \left(\mathsf{cc}_j + n^j_{exp}\right) \leq n^{O(\delta)} \cdot (k^{2\delta} + 1)\end{aligned}$$

- Prover running time (given the UP witnesses):

$$\begin{aligned}\mathcal{P}\mathsf{time}_j &= \mathsf{poly}(n^j_{exp}, n^j_{imp}, n^j_{hol}, s_j \cdot k^{-\delta}) \\ &= \mathsf{poly}(n, k)\end{aligned}$$

To analyze the last three steps of the protocol, first recall that $s_{\ell'}$ (i.e., the length of the list that represents of the final set $S_{\ell'}$) has size smaller than n^δ. Thus, Step (4) adds an additional $s_{\ell'} \cdot (|y_1| + |\mathcal{T}_1|) \leq O(|\mathbb{H}|) \cdot n^{O(\delta)} = n^{O(\delta)}$ communication. The communication complexity of the constant-round HIP in Step (4b) and Step (5) is $n^{O(\delta)} \cdot n^{\delta+o(1)} \leq n^{O(\delta)}$.

As for the verification time, constructing $\langle S_1 \rangle$ takes $\log k$ time (as the circuit $C_{S_1} : [k] \to [k]$ which describes the set $[k]$ is the circuit for $\mathtt{id}$). Constructing $\langle \psi_1 \rangle$ takes $O(\log n)$ time, as explained in Section (4.4.1). Moreover, generating the set $S_{\ell'}$ and the circuit $\psi_{\ell'}$ and evaluating $\psi_{\ell'}$ takes $(q_{\ell'} \cdot n^{\delta+o(1)}) + \tilde{O}(n^{\ell'}_{imp}) \leq s_{\ell'} \cdot k^{-\delta} \cdot n^{\delta+o(1)} + \tilde{O}(s_{\ell'} \cdot n^{O(\delta)}) = O(|\mathbb{H}|) \cdot k^{-\delta} \cdot n^{\delta+o(1)} + \tilde{O}(|\mathbb{H}| \cdot n^{O(\delta)}) = n^{O(\delta)}$ time, using Lemma 4.4, as this is the size of their circuits. The verification time of the constant-round HIP of Steps (4b) and (5) takes $n^{O(\delta)} \cdot (n^{\delta+o(1)} + n_h + s_{\ell'} \cdot (|y_1| + |\mathcal{T}_1|)) \leq n^{O(\delta)}$ time, as the circuits on which these protocols run are $n^{\delta+o(1)}$-succinct.

Since $\ell' \leq 1/\delta + 1$ and the protocol $(\mathcal{P}_{reduction}, \mathcal{V}_{reduction})$ has the same number of rounds as the HIPP, which has $O(1/\delta)$ rounds, we get that the protocol $(\mathcal{P}_i, \mathcal{V}_i)$ has $\ell' = O(1/\delta^2)$ rounds. The depth of the circuit for Step (4b) is $O(1)$ independent of δ, where the depth of the circuit for Step (5) is $O(1/\delta)$. Thus, steps (4b) and (5) take additional $O(1/\delta)$ and $O(1/\delta^2)$ rounds, respectively, and the parameters stated in the theorem's statement follow.

□

4.4.1 Uniformity Proofs

This last section is devoted for uniformity proofs that were deferred throughout the proof of the layer-to-layer subprotocol. It starts with proving the uniformity of the initial predicate ψ_1, on which the first execution of the layer-to-layer subprotocol is run. Then, we prove the uniformity of R and of R_S. Next, we define $R_{\ell'}$, the final relation defined in the protocol, and prove its uniformity. Lastly, we construct the circuit that is used in the last step of the protocol, and prove its uniformity.

The predicate ψ_1 and its uniformity. As already noted above, the predicate of the first recursive call, i.e., ψ_1, has to "represent" the input claim (r_i, v_i). Namely, it must hold that

$$\psi_1\left((y_p, \mathcal{T}_p)_{p\in[k]}\right) = 1 \iff \mathsf{LDE}(w_i)[r_i] = v_i.$$

Therefore, we must carefully define $\langle\psi_1\rangle$ and make sure that it describes a predicate ψ_1 that is indeed enough uniform. Notice that the first part of these pairs, namely, $(y_p)_{p\in[k]}$, is exactly the current layer i on which the layer-to-layer subprotocol works: namely, w_i. Since the input claim (r_i, v_i) is only about w_i, and not about the tableau $(\mathcal{T}_p)_{p\in[k]}$, the predicate ψ_1 ignores the second part in each pair $(y_p, \mathcal{T}_p)_{p\in[k]}$. Thus, to ease notation, we assume that it only gets w_i as input.

Fix the input length $|w_i| = N$. We recall that $r_i \in \mathbb{F}^m$ (where here $m = m(N) = \lceil \log_{|\mathbb{H}|}(N)\rceil$, such that $N \equiv |\mathbb{H}|^m$) uniquely defines the locations that are to be read in the input in order to find the value of its low degree extension at this point (see Equation (3)). Moreover, it defines the coefficients of each summand (see Equation (2)). Using Proposition 3.10, if we take $\hat{\tau}_i$ to be the circuit that computes the coefficient of $i \in \mathbb{H}^m$, then $\hat{\tau}_i : \mathbb{F}^m \to \mathbb{F}$ is an $O(\log n)$-highly uniform arithmetic circuit of size $O(m \cdot |\mathbb{H}|^2)$.

First, we describe $C_{\mathsf{LDE},(r,v)}$, a circuit that computes the value of the low degree extension of its (length-N) input at point r, and outputs 1 if and only if the result is v. This is an arithmetic circuit over the field $\mathbb{F}$ whose advice string is (r, v). Then, we take the circuit that computes ψ_1 to be $C_{\mathsf{LDE},(r_i,v_i)}$ (i.e., $\langle\psi_1\rangle = (r_i, v_i)$).

The circuit is (logically) divided into 6 layers, including the input and the output. In fact, the circuits $(\hat{\tau}_i)_{i\in[N]}$, that the circuit uses, have more than one layer. However, we think of them as a single one and label them accordingly.

The label of each gate is of the form (t, ℓ), where $t \in \{0,1\}^3$ denotes the layer (notice that there is no layer 0), and $\ell \in \{0,1\}^\mu$ denote the internal label, for $\mu = \log N + \xi$ where ξ is the length of a label in each of the circuits $(\hat{\tau}_i)_{i\in[N]}$ (recall that each of them is of size $O(m \cdot |\mathbb{H}|^2)$). It would be easier to think of labels and indices as integers, thus we often use the notation $[k]_2$ for the μ-bit or $\log N$-bit binary expansion of an integer $k \in [2^\mu]$, and it will be clear from the context which one we refer to. To facilitate the labeling, we allow gates in all layers to be fed by the input layer, although the circuit is *layered* and the required gates from the input layer are moved from layer to layer and labeled appropriately.

- **Layer 1:** The first (input) layer is composed of the N input gates, the $(m+1)$ advice gates, and a gate for the constant 1. For $0 \le i \le N-1$, the i^{th} input is labeled $(001, [i]_2)$. For $0 \le j \le m$, the j^{th} advice gate is labeled $(001, [N+j]_2)$. The last gate is the constant 1, and it is labeled $(001, [N+m+1]_2$.

- **Layer 2:** The second layer locates the N circuits $\hat{\tau}_1, ..., \hat{\tau}_N$ in parallel, each of them is fed by (the m gates that hold) r. If the internal labels of each of them are binary strings $\alpha_0, ..., \alpha_{2^\xi-1}$,

then the j^{th} label of the i^{th} gate, for $0 \leq i \leq N-1$ and $0 \leq j \leq 2^{\xi}-1$, is $(010, [i]_2 \circ \alpha_j)$. We label the output gate of $\hat{\tau}_i$ with $(010, [i]_2)$.

- **Layer 3:** The third layer consists of N multiplication gates of fan-in 2, such that for $0 \leq i \leq N-1$, the i^{th} gate computes the product of the i^{th} input and the i^{th} coefficient (i.e., the result of $\hat{\tau}_i$). We label them in the natural way by $(011, [i]_2)$.

- **Layer 4:** The forth layer has a single addition gate of fan-in N. It is labeled $(100, [0]_2)$ and fed by all of the gates of the third layer.

- **Layer 5:** The fifth layer has a single addition gate of fan-in 2. It is labeled $(101, [0]_2)$ and fed by the gate of the previous layer and by the advice gate that holds v. Notice that since addition and subtraction are the same over extension fields of $\mathbb{GF}[2]$, this result outputs 0 if and only if the value of the low degree extension of the input at r is v.

- **Layer 6:** The sixth (output) layer has a single addition gate of fan-in 2. It is labeled $(110, [0]_2)$ and fed by the gate of the previous layer and by the constant 1.

Notice that the length of the advice string to the circuit that computes ψ_1 is $((m+1) \cdot \log(|\mathbb{F}|))$, since we measure the length of the advice string in *bits*, and this term is bounded by $O(\log n)$. Moreover, the fan-in for multiplication gates in $(\hat{\tau}_i)_{i \in [N]}$ is at most $|\mathbb{H}| = n^{\delta}$, as can be seen in Equation (2). Since the circuit only adds a multiplication gate of fan-in 2, its overall fan-in is n^{δ} as well. Regarding the input length N, recall that the input to ψ_1 is w_i, which is of size at most $\mathsf{poly}(n)$, as we assumed that the longest layer is of length $\mathsf{poly}(M) = \mathsf{poly}(n)$, so we get $N = \mathsf{poly}(n)$. Thus, the circuit that computes ψ_1 is of size $N \cdot O(|\mathbb{H}|^2) + O(N) = \mathsf{poly}(n)$, and this implies that ψ_1 is computed by an $\#\mathsf{AC}^0_{\mathbb{F}, f_{in}}$ circuit, where $f_{in} = O(n^{\delta})$. The depth of the circuit is independent of δ, since this is true w.r.t. $(\hat{\tau}_i)_{i \in [N]}$.

Next, we turn to prove its uniformity. Our goal is to construct the adjacency predicate and incidence function of the circuit, and to prove that they are computable by arithmetic circuits over $\mathbb{GF}[2]$ with size and degree at most $O(\log n)$.

First, followed by Proposition 3.10, there exists arithmetic circuits ϕ^{τ}_{adj} and ϕ^{τ}_{incd} that compute the adjacency predicate and incidence function of each τ_i in degree $O(\log n)$. We take

$$\phi_{adj} : \{0,1\}^{3+\mu} \to \{0,1\}$$

to be the circuit's adjacency predicate, and

$$\phi_{incd} : \{0,1\}^{3+\mu} \times \{0,1\}^{\log N} \to \{0,1\}^{3+\mu}$$

to be the circuit's incidence function. We always assume that the functions should return 0 or $0^{3+\mu}$ for all possible labels that are not explicitly mentioned. We explicitly detail the values they compute, and then prove that indeed these values are computable in arithmetic circuits with the claimed properties.

For $0 \leq i \leq |\mathbb{F}|-1$ and $0 \leq j_1, j_2 \leq 2^{\xi}-1$, it should hold that

$$\phi_{adj}\left((010, [i]_2 \circ \alpha_{j_1}), (001, [i]_2 \circ \alpha_{j_2})\right) = \phi^{\tau}_{adj}(\alpha_{j_1}, \alpha_{j_2}).$$

In addition, for $0 \leq i \leq |\mathbb{F}| - 1$,

$$\begin{aligned}
&\phi_{adj}\left((100, [0]_2), (011, [i]_2)\right) = 1,\\
&\phi_{adj}\left((101, [0]_2), (100, [0]_2)\right) = 1,\\
&\phi_{adj}\left((101, [0]_2), (001, [N+m]_2)\right) = 1,
\end{aligned}$$

and

$$\begin{aligned}
&\phi_{adj}\left((110, [0]_2), (101, [0]_2)\right) = 1,\\
&\phi_{adj}\left((110, [0]_2), (101, [N+m+1]_2)\right) = 1.
\end{aligned}$$

For $0 \leq i \leq |\mathbb{F}| - 1$, $0 \leq j \leq 2^{\xi} - 1$ and $k \in [\log N]$, it should hold that

$$\phi_{incd}\left((010, [i]_2 \circ \alpha_j), [k]_2\right) = \phi^{\tau}_{incd}(\alpha_j, [k]_2).$$

In addition, for $0 \leq i \leq |\mathbb{F}| - 1$,

$$\phi_{incd}\left((011, [i]_2), [1]_2\right) = (010, [i]_2),$$

and

$$\phi_{incd}\left((011, [i]_2), [2]_2\right) = (001, [i]_2).$$

Moreover, we relabel the $\alpha_1, ..., \alpha_{2^{\xi}}$ that correspond to the input to the circuits that form the second layer such that the j^{th} input to τ_i, for $0 \leq i \leq N - 1$ and $0 \leq j \leq 2^{\xi} - 1$, is $(001, [N + j]_2)$ (they all have the same input, that is, the coordinate r).

Now, we show that ϕ_{adj} and ϕ_{incd} are computable by arithmetic circuits of logarithmic size and degree. Except for running ϕ^{τ}_{adj} and ϕ^{τ}_{incd}, notice that the only operation that they perform is comparing two binary strings of length at most $O(\log N) = O(\log n)$. Comparing y and z by an arithmetic circuit over $\mathbb{GF}[2]$ can be done as follows:

$$\prod_{i \in [|y|]} \left((1 - y[i]) + z[i]\right) \cdot \left(y[i] + (1 - z[i])\right),$$

while recalling that subtraction is the same as addition. The size and the degree of this circuit is linear in its input length. Since the inputs are of length $O(\log n)$, we conclude that ψ_1 has an $O(\log n)$-succinct $(\log n)$-highly uniform description $\langle \psi_1 \rangle$ and that it is computed by an $\#\mathsf{AC}^0_{\mathbb{F}, f_{in}}$ circuit, where $f_{in} = O(n^{\delta})$.

The relation R and its uniformity. We refer the reader to the discussion at the beginning of Section 4.4 about the chosen structure for R.

Claim 4.7. *The relation R described in Equation (6) is computable by an ensemble of n^{δ}-succinct and $O(\log n)$-highly uniform AC^0 circuits.*

Proof. To simplify notation, we use h in place of h_i throughout this proof. We construct a constant-depth AC^0 circuit $C = C_n$ that computes R. Following Definitions 3.15 and 3.17, we prove that there exists an n^{δ}-bit long string $\langle R \rangle$ and a Turing machine M such that on input $(n, \langle R \rangle)$, the machine runs in time $n^{\delta + o(1)}$ and outputs an arithmetic circuit over $\mathbb{GF}[2]$ of size $n^{\delta + o(1)}$ and degree $O(\log n)$ that computes C's adjacency predicate.

We interpret the binary vector $\mathcal{T}$ in the standard form of a tableau (see, e.g., Theorem 2.21 in [Gol08]): a $P \times P = |\mathcal{T}|$ array, for some polynomial $P = P(\kappa)$, where each row represents a configuration of the Turing machine that computes $h(z)$ (we discuss this machine next), and P represents the time of this computation. W.l.o.g., suppose that the machine is a one-tape Turing machine. Taking Q to be the finite set of states, each such configuration can be encoded by P pairs, where each pair contains the content $\sigma \in \{0,1\}$ of a cell of the working tape, and a symbol $q \in Q \cup \{\perp\}$ that indicates either a state of the machine or the fact that the machine is not located in this cell (in fact, we consider some binary representation of these symbols). The rows are ordered from top (the first configuration) to bottom. With the exception of the first row, the values of the entries in each row are determined by (up to) three entries of the row just above it, which represents the previous configuration, where this determination reflects the function h.

Notice that if h were computable by some *fixed* Turing machine $\tilde{M}_h$ that on input z returns $h(z)$, then building a (highly-uniform) circuit asserting that $\mathcal{T}$ is the correct tableau for the computation $h(z) = y$ given $(z, y, \mathcal{T})$ would be immediate: the circuit would simply use the constant-size transition function of $\tilde{M}_h$ in order to verify the consistency of each two consecutive configurations, as done in the Cook-Levin theorem. In fact, our computation would be even "easier", since the circuit gets the tableau of the computation as input instead of computing it by itself, and that is why this circuit is in AC^0.

However, since h is a UOWHF, chosen from a family $\mathcal{H}$ using a randomly chosen key k_h, there is no such "keyless" machine. Nevertheless, one of the properties of $\mathcal{H}$ promises that there exists a machine $M_{\mathcal{H}}$, such that on input (k_h, z), outputs $h(z)$. Followed by the foregoing argument, this means that there exists a (highly-uniform) circuit $C_{\mathcal{H}}$, such that on input $(k_h, z, y, \mathcal{T})$, outputs 1 if and only if $\mathcal{T}$ is the correct tableau for the computation $h(z) = y$. We use this circuit in order to construct the circuit C, which gets as input an advice string k_h, in the following manner: C emulates $C_{\mathcal{H}}$, only that the key k_h will be "hard-wired" into $C_{\mathcal{H}}$, and C will also check that the bits in the locations of the key (which is a part of its input) are indeed k_h (i.e., equal to the bits of the advice that is hard-wired to the circuit). The full details follow.

Structure. First, we describe the circuit $C_{\mathcal{H}}$. Take δ to be the transition function of $M_{\mathcal{H}}$ as described above, and assume it is given in the form of a lookup table: it is a constant-size collection of 4-tuple entries, where each entry consists of a pair. For $\sigma_i \in \{0,1\}, q_i \in Q \cup \{\perp\}$, a 4-tuple $(\sigma_1, q_1), (\sigma_2, q_2), (\sigma_3, q_3), (\sigma_4, q_4)$ is in the collection if and only if

$$\delta\big((\sigma_1, q_1), (\sigma_2, q_2), (\sigma_3, q_3)\big) = (\sigma_4, q_4),$$

where δ considers the current machine state, the bit in the tape at the location of the reading head, a new value for the location just read from the tape, and a direction to move the reading head.

Consider the set of all of the different 4-tuple entries in $\mathcal{T}$ (there are $|\mathcal{T}|$ of them; each entry of the tableau appears exactly once as the middle pair (σ_2, q_2)). Given h's key k_h, if $\mathcal{T}$ is indeed the unique tableau of the computation $h(z) = y$, then this set should be a subset of the collection induced by the lookup table: namely, each 4-tuple of tableau entries should appear in the lookup table. Hence, checking the validity of $\mathcal{T}$ boils down to checking that all of the 4-tuples of tableau entries are included in the collection. Moreover, we must check that the first configuration of $\mathcal{T}$ contains z and that the last configuration contains y.

Formally, the circuit $C_{\mathcal{H}}$ consists of two central components, each one of them is a constant-depth and polynomial-size AC^0 circuit. The first component compares z and y to the first and the last

configurations of $\mathcal{T}$. The comparison to z is done using n_{in} conjunction gates and $2n_{in}$ disjunction and negation gates, such that all of the gates of the same type are located in the same layer (i.e., in parallel). The i^{th} gate is connected to the i^{th} bit of the first configuration and the i^{th} bit of z in the following manner: denoting the former by b and the latter by b', the result of each comparison is given by a constant-size circuit C_{EQ} that computes $((b \vee \neg b') \wedge (\neg b \vee b'))$. The comparison to y is done analogously.

The second component is the one that checks the local consistency of each two consecutive configurations. Recall that δ can be described as a constant-size circuit, that is the disjunction of all of the tuples in the lookup table. Denote this circuit by C_δ. For each entry (i, j) of the array that $\mathcal{T}$ represents, excluding $i = 1$, the component connects the gates that match the entries $(i-1, j-1), (i-1, j), (i-1, j+1)$ to a copy of C_δ (or only two entries, in case j is the first or the last column). This means that there are $O(|\mathcal{T}|)$ such copies, and all of them are located "in parallel", i.e., in the same layers.

The outputs of all of these conjunction gates of the first component and the circuits C_δ of the second component are connected to a conjunction gate, of fan-in $n_{in} + n_{out} + O(|\mathcal{T}|)$. The output of the circuit is the output of this gate.

It is clear that the circuit described above is an AC^0 circuit. The circuit C merely adds a comparison between the key that it got as input, that is, the advice string, and the input of $C_{\mathcal{H}}$. These comparisons are done by the circuit C_{EQ}. Finally, C connects the result of $C_{\mathcal{H}}$ and all of the comparisons with a conjunction gate, and that is its output. It is evident that the depth of the circuit is independent of δ.

Uniformity. Take k_h to be h's key and recall that C is given the advice string $\langle R \rangle = k_h$. The length of k_h is κ, and since we took $\kappa = n^\delta$, we get that $|\langle R \rangle| = |k_h| = n^\delta$. Denote by $(0, [1]_2), ..., (0, [|k_h|]_2)$ the labels of the input gates that correspond to k_h (notice that these labels are in binary).

The machine M builds the adjacency predicate of C, denoted ϕ, in the following manner. For comparing between the i^{th} bit of the gates that correspond to the locations of the key in $C_{\mathcal{H}}$, that we label with $(1, [i]_2)$ for $i \in [|k|]$, to the i^{th} bit of the advice string, that is $(0, [i]_2)$ for $i \in [|k_h|]$ (of course, $|k| = |k_h|$), it should hold that:

$$\phi\left((0, [i]_2), (1, [i]_2)\right) = 1.$$

The rest of the labels of $C_{\mathcal{H}}$ are labeled as a tuple of three binary strings (i, j, k), where the first two indices specify an input gate or a gate of the first component (i.e., one of the gates used for the comparison), and the third index specifies a gate in the circuit C_δ. In particular, this means that the first two indices are zero for gates of C_δ, and vice versa. We make sure that these labels are disjoint from the labels that we already fixed for the key gates.

The circuit for the first component, denoted ϕ_1, ignores the third index, and for every pair of gates $(i_1, j_1), (i_2, j_2)$, outputs 1 if and only if they are either (1) two pairs from the set $(i-1, j-1), (i-1, j), (i-1, j+1), (i, j)$, or (2) $j_1 = j_2$ and i_1, i_2 indicate that one of them is a gate that corresponds to the first or the last row of the tableau, and the other is a gate that corresponds to the input z or y, respectively. Notice that ϕ_1 only compares indices or checks if they are consecutive. Moreover, ϕ_1 returns 0 when it gets as input $(i_1, j_1) = (i_2, j_2) = \vec{0}$. This is important, since when it gets labels of gates in C_δ and ignores the third index, it is left with two sequences of zeros.

We take ϕ_δ to denote C_δ's adjacency predicate. Once again, by the way we defined the labels for $C_{\mathcal{H}}$, we make sure that ϕ_δ returns zero when it gets $k_1 = k_2 = \vec{0}$.

Finally, for any two gates $(i_1, j_1, k_1), (i_2, j_2, k_2)$, we define ϕ such that

$$\phi((i_1, j_1, k_1), (i_2, j_2, k_2)) = 1 - (1 - \phi_1((i_1, j_1), (i_2, j_2))) \cdot (1 - \phi_\delta(k_1, k_2)),$$

when recalling that subtraction and addition are the same over $\mathbb{GF}[2]$. All in all, these operations can all be done using an arithmetic circuit over $\mathbb{GF}[2]$ whose size and degree are linear in its input length, as comparison can be performed in linear size (see, e.g., the uniformity proof of the predicate ψ_1 at the beginning of Section 4.4.1, or Claim 3.22, for full details). All of the labels are of length $O(\log(|\mathcal{T}| + |k_h|)) = O(\log n)$, hence, the adjacency predicate is computable by a degree-$O(\log n)$ arithmetic circuit. We conclude that C is n^δ-succinct and $O(\log n)$-highly uniform AC^0 circuit.

□

The relation R_S and its uniformity. The following claim is true with respect to any NP relation R. However, in order to facilitate the notation, one may think of the relation R as defined in Equation (6), where the explicit input x is h_i and each witness y_p is the augmented witness $(y_p, \mathcal{T}_p)$. We comment that we do not assume any succinctness condition about the set S in the following claim: although the inputs to R_S, namely, $(y_p)_{p\in S}$ and $(z_p)_{p\in S}$, are ordered by (the list that represents) the set S, the circuit that computes R will operate on this sequence as is, in the order that it is given. This order does not change its functionality.

Claim 4.8. *Let $\delta \in (0, 1]$ be a constant, and let $\mathbb{H} \subseteq \mathbb{F}$ be (ensembles of) extension fields of $\mathbb{GF}[2]$, where $|\mathbb{H}_n| = \Theta(n^\delta)$ and $|\mathbb{F}_n| = \mathsf{poly}(n)$.*

Suppose R is an NP relation computable by an ensemble of n^δ-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuits, whose witnesses' length is n_{imp}, and let R_S be as described in Equation (7). Assume that $S \subseteq [k]$ is a set of size $q \leq \mathsf{poly}(n)$, and that $\langle\psi\rangle$ is an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform description for an arithmetic predicate $\psi : \{0,1\}^{q\cdot n_{imp}} \to \mathbb{F}$, computed by a $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit where $f_{in} = O(n^\delta)$.

Then, R_S is computable by an ensemble of $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits, where $f_{in} = O(n^\delta)$.

Proof. Let R, R_S, S and ψ be as described in the hypothesis. We construct a circuit $C = C_n$ that computes R_S, such that the following holds:

- **Structure:** C is an $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit with $f_{in} = O(n^\delta)$;
- **Uniformity:** C is an $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform arithmetic circuit.

The circuit C is composed of two parts: the component that checks the conditions of R, denoted $C_{\wedge R}$, and the component that checks the condition of ψ, denoted C_ψ. First, we build the circuit $C_{\wedge R}$. We use Theorem 3.18 in order to approximate the circuits that form it, as they are Boolean, and get an arithmetic circuit with a small multiplication gates fan-in. Then, we construct C_ψ, which is arithmetic, and put the components together to get C. We prove that the structure and the uniformity of C are as stated above.

Structure. First, we take C_R to be the n^δ-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit that computes the relation R (actually, in Claim 4.7, we proved that C_R is an AC^0 circuit). Using Theorem 3.18 with $\lambda(n) = \log^2 n$, there exists a probabilistic arithmetic circuit $\widehat{C}_R$ in $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$, where $f_{in} = O(\log(|C_R|) + \log^2 n) = O(\log^2 n)$ (and in particular, $f_{in} = O(n^\delta)$). The circuit $\widehat{C}_R$ approximates C_R, and they agree on every input with all but a $2^{-\log^2 n}$ probability. It inherits the uniformity condition of C_R, and we use this fact in what follows.

Next, we define $C_{\wedge R}$ to be the circuit that runs in parallel q copies of the circuit $\widehat{C}_R$. The copies are relabeled in the natural way, by adding a prefix i to each label in the i^{th} copy of $\widehat{C}_R$. The i^{th} copy gets as input (x, z_i, y_i), according to the order in the list that represents the set S, which is the way that both the implicit and the holographic inputs to R_S are ordered. We wish to stress that some of these copies may be identical, in the case that $q > |S|$. The output of $C_{\wedge R}$ (which is of length q) is the q results of these copies. Since $q \leq \mathsf{poly}(n)$, it follows that $C_{\wedge R}$ is in $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ as well, and with $f_{in} = O(n^\delta)$.

For C_ψ, we use the hypothesis and simply take it to be the $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit that computes ψ, where $f_{in} = O(n^\delta)$. We assume that the labels of C_ψ and $\widehat{C}_R$ are disjoint, and otherwise we relabel C_ψ. In both cases, we add the prefix $[q+1]_2$ to each label in C_ψ.

Finally, we define the output of C to be the outputs of C_ψ and $C_{\wedge R}$. This means that on input x to R_S, $x \in R_S \iff C(x) = 1^{q+1}$. We conclude that C is in $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$, where $f_{in} = O(n^\delta)$. Assuming that the depth of C_ψ is independent of δ (which is the case in all usages), the depth of C is independent of δ as well.

Uniformity. Through this section, we extensively use the different representations for circuits and the definitions for uniformity. We refer the reader to Section 3.3 for more details on these. We stress that the size of an adjacency predicate or an incidence function is measured with respect to the *circuits' index* n, rather than to their input length, which may be logarithmic in n (as it represents a label in a circuit of size $\mathsf{poly}(n)$).

For $s = n^{\delta+o(1)}$, we prove that there exists a Turing machine M such that on input $(n, \langle\psi\rangle \circ \langle R\rangle)$, runs in time $(s^{1+o(1)} \cdot n^{o(1)})$ and outputs two arithmetic circuits over $\mathbb{GF}[2]$ of size $(s^{1+o(1)} \cdot n^{o(1)})$ and degree $O(\log n)$ that compute C's adjacency predicate for the circuit's addition gates and the incidence function for the circuit's multiplication gates.

By the assumption, C_R is an n^δ-succinct and $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit. As mentioned earlier, $\widehat{C}_R$ inherits the uniformity condition of C_R, and that implies that $\widehat{C}_R$ is an n^δ-succinct and $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit as well.

We take $(\langle\psi\rangle \circ \langle R\rangle)$ to be C's advice string. The machine M works as follows: First, it runs the machines that output the adjacency predicates and incidence functions of $\widehat{C}_R$ and C_ψ, where it uses the advice $(n, \langle R\rangle)$ for the first machine and the advice $(n, \langle\psi\rangle)$ for the second. Then, given the two arithmetic circuits (see Definition 3.17) that compute the adjacency predicates for the addition gates of $\widehat{C}_R$ and C_ψ, the machine M defines the arithmetic circuit that computes the adjacency predicate for the addition gates of C in the following manner: On input (u, v), the circuit ignores the first $\lceil \log q \rceil$ bits of u and of v, and outputs the sum of outputs of the two circuits that compute the adjacency predicates of $\widehat{C}_R$ and C_ψ, where their inputs are the suffixes of u and v. Here, we used the convention that the circuit computing the incidence function outputs 0 when u or v are not existing labels, and that we took the labels to be disjoint. Notice that, logically, this addition gate serves as a disjunction gate, as these arithmetic circuits are over $\mathbb{GF}[2]$.

The size of the circuit that computes the adjacency predicate for the addition gates of $\widehat{C}_R$ is $s = n^{\delta+o(1)}$, and of C_ψ is s as well, since $|\langle\psi\rangle| = s$ and $(s^{1+o(1)} \cdot n^{o(1)}) = n^{\delta+o(1)} = s$. Thus, the size of the circuit that computes the adjacency predicate of C is $n^{\delta+o(1)}$. By the construction, it is evident that M works in linear time in the size of the these circuits. Furthermore, the degree of both these circuit is only increased by 1, since we only added a single multiplication gate, and since the degree of both of them is $O(\log n)$ and they are connected in parallel.

M works analogously for the multiplication gates' incidence function of C. On input (u, i), where u is a gate and i is an index, the circuit computing the incidence function ignores the first $\lceil \log q \rceil$ bits of u, and outputs the sum of the outputs of the two circuits for the incidence function of $\widehat{C}_R$ and C_ψ (there are $|u|$ multiplication gates, one per each output bit of the circuits), where their inputs are the suffix of u together with i. Here, we used the convention that the circuit computing the incidence function outputs $0^{|u|}$ when u is not an existing label, and that we took the labels to be disjoint.

By the same argument, the size of this circuit is $n^{\delta+o(1)}$, and its degree is $O(\log n) + 1$. Followed by Definition 3.17, we conclude that C an $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit, and the claim follows.

□

The relation $R_{\ell'}$ and its uniformity. We refer the reader to the end of Section 4.3.1 for some high-level details about the role of $R_{\ell'}$. For Step (4b) of the protocol, we define $R_{\ell'}$ on holographic input $(z_p)_{p\in S_{\ell'}}$ and explicit input $(h_i, (\tilde{y}_p, \tilde{\mathcal{T}}_p)_{p\in S_{\ell'}})$, such that the input satisfies the relation if and only if $\forall p \in S_{\ell'}$, $\tilde{\mathcal{T}}_p$ is the correct tableau for the computation $h_i(z_p) = \tilde{y}_p$. In order to justify the usage of Theorem 3.25 on this relation, it must be computable in $s(n)$-succinct and $d(n)$-highly uniform $\mathsf{AC}^0[\oplus]$ with $s(n) \leq n$. However, this is already proved in Claim 4.8: We can view $R_{\ell'}$ as a special case of $R_{S_{\ell'}}$, when taking R to be as in Equation (6) and ψ to be a tautology. We get that $s(n) \leq n^{\delta+o(1)}$ and that $d = O(\log n)$.

Constructing the circuit for Step (5). Recall the goal of the HIP executed in Step (5) of the protocol: It reduces the set of claims $(\chi_j, \theta_j)_{j\in[\ell']}$ about $(z_p)_{p\in[k]}$ to a single claim (r_{i+1}, v_{i+1}) about it. However, each of the input claims (χ_j, θ_j) are about $(z_p)_{p\in S_j}$, namely, on the holographic input when ordered according to *the list that represents it*, rather by the natural order of elements in $[k]$ (see Remark 4.3). On the other hand, the output claim *should* be about the sorted $(z_p)_{p\in[k]}$ (in the natural order), since $z_1, ..., z_k$ form the next layer of the hash tree w_{i+1}. Thus, the circuit we construct in the rest of this section, which computes $(\chi_j, \theta_j)_{j\in[\ell']}$, reduces ℓ' claims about the unsorted holographic input to a single claim about the sorted holographic input.

Formally, recall that the circuit gets as input $(z_p)_{p\in[k]}$, and as an advice the sequence of descriptions $(\langle S_j\rangle)_{j\in[\ell']}$ and the sequence of claims $(\chi_j, \theta_j)_{j\in[\ell']}$. In order to run the constant-round HIP of Theorem 3.25, we need to prove that this circuit is an $n^{\delta+o(1)}$-succinct and $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit with multiplication gate fan-in $f_{in} = O(n^\delta)$. Denote this circuit by C.

For every $j \in [\ell']$, The advice $\langle S_j\rangle$ can be used to build C_{S_j}, an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit of size $n^{\delta+o(1)}$, that maps indices in $[q_{j-1}] = [s_{j-1} \cdot k^{-\delta}]$ (given in binary representation) to elements in S_j. We recall that s_j is the length of the list that represents the set S_j, where $s_j = q_{j-1}$ and $s_1 = q_0 = k$. Thus, $q_{j-1} = s_1 \cdot k^{-(j-1)\cdot\delta} = k^{1-(j-1)\delta}$.

First, for every $j \in [\ell']$, we define the following circuit C_j:

$$C_j = C_{S_1} \circ ... \circ C_{S_{j-1}} \circ C_{S_j}.$$

Logically, this circuit maps indices in S_j to their "original" indices in $S_1 = [k]$. Notice that the list for S_j is of length q_{j-1}, by the way it is defined in the protocol. Thus, $C_j : \{0,1\}^{\log q_{j-1}} \to [k]$, and on input $[r]_2$ for $r \in [q_{j-1}]$, it returns the r^{th} "original" index in the list that represents S_j. Actually, we refer to C_j as the circuit described above after arithmetization (that is performed in the straightforward way, without approximation). Without loss of generality, since each of the circuits C_{S_j} is of size $n^{\delta+o(1)}$, the fan-in for multiplication gates in C_j is upper-bounded by n^{δ}.[19] Regarding the depth of C_j, since it composes j circuits of a constant depth *independent of* δ, as every C_{S_j} satisfies this property (as stated in Theorem 3.26) its depth is $(c \cdot j)$ for a global constant c. However, note that j does depend on δ, and we emphasize that this is only place in this work when we use the term *constant-depth* circuit and the depth depends on δ.

We define the following circuit C_{select} that "selects" z_i, given $z_1, ..., z_k$ and an index i. Although this circuit is arithmetic, it only works on Boolean values, and this fact is crucial in its construction. The circuit works as follows:

- Its input is $i \in \{0,1\}^{\log k}$ and $z_1, ..., z_k \in \{0,1\}^{|z_1| \cdot k}$.
- Its first layer compares between i and all of the elements in $[k]$ (namely, the k binary strings $[1]_2, ..., [k]_2 \in \{0,1\}^{\log k}$), using the following identity for two bits b, b': $(b = b') \equiv ((1-b)+b') \cdot (b + (1-b'))$.
- The second layer shrinks each of the k results to a single bit, by k multiplication gates of fan-in $\log k$. This bit equals 1 if and only if the two strings are equal. This implies that, at this point, only the i^{th} bit is 1, and the rest are 0.
- The third layer uses $k \cdot |z_1|$ multiplication gates of fan-in 2, that operate on the results of the third layer (each one has fan-out $|z_1|$) and the inputs $z_1, ..., z_k$. If we interpret this layer as k sequences of length $|z_1|$, we get that only the i^{th} sequence is the string z_i, and the rest of them are $0^{|z_1|}$.
- The forth (output) layer is the sum of these k results, i.e., $|z_1|$ addition gates of fan-in k. Since a sum of a bit and 0's returns the bit, we get that the circuit's output is exactly the string z_i.

By the construction, it is evident that the circuit C_{select} is an $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit, where $f_{in} = \log k = O(\log n)$.[20] Its depth is 4, i.e., a global constant independent of δ.

Take i_r to be the output of C_j on index $r \in [q_{j-1}]$ (i.e., on $[r]_2$). This means that

$$\forall r \in [q_{j-1}],\ C_{\mathsf{select}}(z_1, ..., z_k, i_r) = z_{i_r}.$$

Therefore, $(z_p)_{p \in S_j} = z_{i_1}, ..., z_{i_{q_{j-1}}}$, which implies that the j^{th} claim (χ_j, θ_j) is about $z_{i_1}, ..., z_{i_{q_{j-1}}}$.

[19] If there exists a gate u in C_j with $f_{in} > n^{\delta}$, split it into n^{δ} gates of the same type of u, with $f_{in} = n^{\delta+o(1)}/n^{\delta} < n^{\delta}$, and connect them with a gate with fan-in n^{δ}, that is also of the same type. Repeat this process for any such u. This increases the size of the circuit by at most a polynomial factor, whereas its depth is at most squared.

[20] The size of each layer i is $\mathsf{poly}(n)$, as explained later in the proof of Theorem 4.2, and thus it must hold that for any $k = k(i)$, $\log k = O(\log n)$.

Let us stop for a moment and take a closer look at what is done so far. For any fixed $j \in [\ell']$, our goal is to check if the claim (χ_j, θ_j) about $(z_p)_{p\in S_j}$ is correct, whereas the input to the circuit C is $(z_p)_{p\in[k]}$. Towards this, our building block is a circuit C_j, that on index r returns the r^{th} element of the list that represents S_j. We denote each such output it by i_r, and use it as input to a circuit that outputs z_{i_r}, namely, the appropriate block of the holographic input. This means that the sequence $z_{i_1}, ..., z_{i_{q_{j-1}}}$ is exactly $(z_p)_{p\in S_j}$, that is, the holographic input when ordered by the list that represents the set S_j.

Thus, feeding the circuit that computes the low degree extension at the coordinate χ_j with this sequence, outputs a value which equals θ_j if and only if the claim (χ_j, θ_j) is a correct claim about $(z_p)_{p\in S_j}$. Take $C_{\mathsf{LDE},(\chi_j,\theta_j)}$ to denote this circuit. The full circuit is described at the beginning of Section 4.4.1, when constructing the predicate ψ_1 (there, it is called $C_{\mathsf{LDE},(r,v)}$). Once again, we comment that the depth of $C_{\mathsf{LDE},(\chi_j,\theta_j)}$ is a global constant independent of δ.

Putting it all together, we denote by C'_j the circuit that is composed of q_{j-1} copies of the circuit C_j as described above, located in parallel, where $q_{j-1} = k^{1-(j-1)\delta}$. The r^{th} copy of C_j is fed with the constant $[r]_2$, for $r \in [q_{j-1}]$. The output of the r^{th} copy, denoted $i_r \in [k]$, is the input to a copy of C_{select} (overall, there are q_{j-1} parallel copies of C_{select} for every j), together with $z_1, ..., z_k$. The output, z_{i_r}, is the i_r^{th} input to $C_{\mathsf{LDE},(\chi_j,\theta_j)}$, and this is how the sequence $z_{i_1}, ..., z_{i_{q_{j-1}}}$ is produced. Hence, the output of the circuit C'_j equals 1 if and only if (χ_j, θ_j) is a correct claim about $(z_p)_{p\in S_j}$.

Next, we use these circuits that compute the correctness of each claim (χ_j, θ_j). For every $j \in [\ell']$, C runs $C'_1, ..., C'_{\ell'}$ in parallel. Its output is $1^{\ell'}$ if and only if the complete set of claims $(\chi_j, \theta_j)_{j\in[\ell']}$ has only correct claims about $(z_p)_{p\in[k]}$.

Using the assumption, the construction and Proposition 3.11, we get that all of the subcircuits of C are $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits where $f_{in} = O(n^\delta)$. Since they all run in parallel, we conclude that C is also an $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit where $f_{in} = O(n^\delta)$. The depth of C is dominated by $C_{\ell'}$, as it is the deepest out of all the circuits C_j, and all other circuits described above, as mentioned, are of constant depth where the constant is independent of δ. Thus, C's depth is $O(1/\delta)$.

Regarding the advice, notice that the length of the advice string for C is

$$\ell' \cdot (|\langle S_1\rangle| + |(\chi_1, \theta_1)|) = \ell' \cdot \left(n^{\delta+o(1)} + (m+1)\cdot \log \mathbb{F}\right) = n^{\delta+o(1)}.$$[21]

Finally, we show that the C is $O(\log n)$-highly uniform. The proof follows along similar lines to the proof of Claim 4.8, thus we only provide a sketch. Each label consists of a tuple of three binary indices (α, β, γ), where each index is given in binary. α indicates the copy of C'_j that the gate belongs to, that is, $\alpha = [j]_2$. Next, β indicates the functionality of the subcircuit that the gate belongs to: whether it is in $C_{\mathsf{LDE},(\chi_j,\theta_j)}$, or in one of the q_{j-1} copies of C_j or C_{select}. The third index γ indicates the internal index of the specific subcircuit. In case that β indicates that the gate belongs to a C_j circuit, γ is interpreted as (γ_1, γ_2) where γ_1 indicates which C_{S_p} the gate belongs to (for $p \in [j]$), where γ_2 is the internal label of C_{S_p}.

A careful examination of all of the conditions that the adjacency predicate or incidence function should satisfy yields that there is only a constant number of them. For instance, connecting an output of some C_j to be the input of some C_{select} only requires labeling them with the same γ_1, and then comparing it. Then, although there are up to k copies of them per each C'_j, they only induce one condition, that is, comparing a substring of their labels. Moreover, these conditions

[21] The previous footnote applies here as well: bounding the size of m and ℓ' in terms of n is only formally justified in Theorem 4.2, where the length of each layer is set.

(that connect between different subcircuits) boil down to comparing labels, checking that they are consecutive, or computing the binary representations of field constants.[22] Since each of these operations can be done by an arithmetic circuit over $\mathbb{GF}[2]$ whose size and degree are linear in its input length, and since we can connect them in parallel and only increase their degree by 1, we are left with proving that each of the circuits C_{S_p}, C_{select} and $C_{\mathsf{LDE},(\chi_j,\theta_j)}$ satisfy this uniformity condition.

First, by the assumption, we know that C_{S_p} is $O(\log n)$-highly uniform. Secondly, a full proof for the uniformity of C_{select} can be found in Claim 3.22. Lastly, at the beginning of Section 4.4.1, when constructing the predicate ψ_1, we prove that $C_{\mathsf{LDE},(\chi_j,\theta_j)}$ has an $O(\log n)$-succinct $(\log n)$-highly uniform description (that is, (χ_j,θ_j)). We conclude that C is $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$.

4.5 Proof of Theorem 4.2

Completeness. The perfect completeness follows immediately from the protocol description, as well as from the perfect completeness of $(\mathcal{P}_i,\mathcal{V}_i)$.

Soundness. For the soundness condition, fix an input $\big(w,(y,h)\big) \notin \mathcal{L}_{\mathsf{HHR}}$ and assume that after interacting with a cheating prover $\mathcal{P}^*_{\mathcal{L}_{\mathsf{HHR}}}$, the verifier $\mathcal{V}_{\mathcal{L}_{\mathsf{HHR}}}$ does not reject and outputs a pair (r,v). Let S denote the event that this claim is true, i.e.,

$$\Pr[S] = \Pr[\mathcal{V}_{\mathcal{L}_{\mathsf{HHR}}} \text{ does not reject and } \mathsf{LDE}(w)[r] = v].$$

We wish to bound $\Pr[S]$. Denote by y' the correct hash root of $\mathsf{LDE}(w)$. Since $y \neq y'$, we get that

$$\Pr\left[\mathsf{LDE}(y)[r_1] = \mathsf{LDE}(y')[r_1]\right] \leq \frac{(|\mathbb{H}|-1)\cdot m}{|\mathbb{F}|} = O(n^\delta)/|\mathbb{F}|,$$

followed by Lemma 3.8, that states that every two distinct polynomials of (total) degree $\leq (|H|-1)\cdot m$ over $\mathbb{F}$ agree in at most $\frac{(|\mathbb{H}|-1)\cdot m}{|\mathbb{F}|}$ points. Denote this event by C.

If we assume $(\neg C \wedge S)$, we get that $v_1 = \hat{w}_1[r_1] = \mathsf{LDE}(y)[r_1] \neq \mathsf{LDE}(y')[r_1]$ but $\mathsf{LDE}(w)[r] = v$. In other words, the first claim (r_1,v_1) is false *with respect to* w, while the last claim $(r_{\ell+1},v_{\ell+1}) = (r,v)$ is true. Recall that the $\mathcal{L}_{\mathsf{HHR}}$ protocol consists of ℓ phases. Each phase consists of a single $(\mathcal{P}_i,\mathcal{V}_i)$ run, whose soundness error is $n^{O(\delta)}/|\mathbb{F}|$.

Following the same logic of the proof of Lemma 4.4, if the verifier does not reject, then there must exist a phase $i \in [\ell-1]$ such that the input claim (r_i,v_i) is correct with respect to w and the output claim (r_{i+1},v_{i+1}) is false. This means that the soundness guarantee of $(\mathcal{P}_i,\mathcal{V}_i)$ is broken, and thus this event happens with probability at most $n^{O(\delta)}/|\mathbb{F}|$. Taking a Union Bound and recalling that $\ell = O(1/\delta)$ yield that:

$$\Pr[S] = \Pr[(S\wedge C)\vee(S\wedge\neg C)] \leq \Pr[S\wedge C] + \Pr[S\wedge\neg C] \leq \frac{O(n^\delta)}{|\mathbb{F}|} + \sum_{i=1}^{\ell-1}\frac{n^{O(\delta)}}{|\mathbb{F}|} \leq n^{O(\delta)}/|\mathbb{F}|.$$

[22] By that, we refer to the constant $r \in [q_{j-1}]$ that feeds the r^{th} copy of C_j. The circuit needs to wire the constants 0 and 1 to the appropriate gate of $[r]_2$. The exact same issue is fully handled in Claim 3.22, when proving the uniformity requirements for Layer 6. The depth required for preforming this mapping is a constant independent of δ.

Complexity. The stated complexity follows from the complexity of the $(\mathcal{P}_i, \mathcal{V}_i)$ protocol. The protocol is run for $\ell = O(1/\delta)$ times, since we assumed that $|w| = M = \mathsf{poly}(n)$ (see Section 4.1). Since $(\mathcal{P}_i, \mathcal{V}_i)$ has $O(1/\delta^2)$ rounds, we overall get $O(1/\delta^3)$ rounds.

The input size is larger in each iteration, and this is expressed via the parameter $k = k(i)$ (the rest of the parameters n_{in}, n_{out}, n_h and $|\mathcal{T}|$ are the same for every i). Therefore, we consider the length of the longest input (that is, the input to $(\mathcal{P}_\ell, \mathcal{V}_\ell)$), where $k(\ell) = |w_{\ell+1}|/n_{in} = |\mathsf{LDE}(w)|/n^{2\delta}$.

We know that $|\mathsf{LDE}(w)| = \mathsf{poly}(M)$, and we assumed that $M = \mathsf{poly}(n)$. In fact, we consider the binary representation of $\mathsf{LDE}(w)$ (see Remark 3.7), which is of length $\mathsf{poly}(M) \cdot \log(|\mathbb{F}|)$, and in particular $\mathsf{poly}(n)$.

Thus, $k \leq \mathsf{poly}(n)$ and the prover runs in time $\ell \cdot \mathsf{poly}(n, k) = \mathsf{poly}(n)$. As for the verification time and the communication complexity, we get that $n^{O(\delta)} \cdot (k^{2\delta} + 1) = n^{O(\delta)}$. We note that finding v_1 in Step (1) adds $n_{out} \cdot O(n^{2\delta}) = n^{O(\delta)}$ time, using Proposition 3.11.

□

5 Flat-GKR

After establishing the protocol for HHR, we are ready to state and prove the main result: a constant-round argument system, whose security only relies on the existence of one-way functions, where the communication and the verification time grow linearly with the depth of the circuits computing the language. As mentioned at the end of the technical overview (Section 2), we refer to this proof system as "flat-GKR", as it essentially "flattens" the GKR protocol, and performs its consistency checks between circuit layers *in parallel*, instead of *sequentially*. We begin with a formal theorem statement.

Theorem 5.1 (Theorem 1.1, restated)**.** *Assume one-way functions exist and let $\delta \in (0, 1]$ be a constant. For every language $\mathcal{L}$ that is computable by log-space uniform circuits of fan-in 2, depth $D(n)$ and polynomial size, there is a constant-round doubly-efficient argument system as follows. The protocol is public-coin and has perfect completeness and negligible soundness error. Taking n to be the input length, the protocol's complexities are:*

- *constant round complexity $O(1/\delta^3)$,*
- *communication complexity $D(n) \cdot n^{O(\delta)}$,*
- *verifier runtime $(D(n) + n) \cdot n^{O(\delta)}$. Moreover, if the verifier is given oracle access to the low-degree extension of the input, then its running time is only $D(n) \cdot n^{O(\delta)}$,*
- *the honest prover runs in $\mathsf{poly}(n)$ time,*
- *assuming the existence of one-way functions, the protocol is sound against malicious cheating provers running in time $\mathsf{poly}(n)$.*

Denote by $(C_n)_{n\in\mathbb{N}}$ the ensemble of Boolean circuits that compute $\mathcal{L}$. In our proof, we use the following fact (see [Gol17, Chapter 3]): Any log-space uniform circuit C_n of depth $D(n) \geq \log n$ can be emulated by a circuit of size $\mathsf{poly}(n)$ and depth $O(\log n) \cdot D(n)$, such that the adjacency predicate of the new circuit can be computed by a $\mathsf{polylog}(n)$-size formula that can be constructed in $\mathsf{polylog}(n)$-time. We emphasize that the input to the new circuit is the same as the input to C_n,

and of the same length n (the description of C_n is hard-wired to the new circuit as advice). Thus, w.l.o.g., we assume that C_n itself satisfies this property, and denote its depth by $d(n) = D(n) \cdot \log n$, and its size by $s(n) = \mathsf{poly}(n)$.

Following [GKR15], we assume that the circuit is an arithmetic circuit over the field $\mathbb{F}$. This is possible, as any Boolean circuit can be converted to an arithmetic circuit while increasing the size and the depth of the circuit by at most a constant factor. Moreover, as explained in Section 3.3, we assume without loss of generality that it is a layered arithmetic circuit of fan-in 2, and that the size of each layer is $s = s(n)$.

For each $1 \leq i \leq d$, we let $V_i : [s] \to \mathbb{F}$ be a function that gets a gate $j \in [s]$ and returns the value of the j^{th} gate of the i^{th} layer of the circuit on input x. As before, we find $m = m(s) = \lceil \log_{|\mathbb{H}|}(s) \rceil$ such that $[s] \equiv |\mathbb{H}|^m$, and for each $1 \leq i \leq d-1$, we let

$$\hat{V}_i = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}(V_i) : \mathbb{F}^m \to \mathbb{F}.$$

By Proposition 3.11, the time it takes to compute $\hat{V}_i$ is $|\mathbb{H}|^m \cdot O(m \cdot |\mathbb{H}|^2) = s \cdot O(m \cdot |\mathbb{H}|^2)$.

Since the d^{th} layer, that is, the input layer, is shorter (as $n \leq s$), we find $m' = m'(n) = \lceil \log_{|\mathbb{H}|}(n) \rceil$ such that $[n] \equiv |\mathbb{H}|^{m'}$, and let

$$\hat{V}_d = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}(V_d) : \mathbb{F}^{m'} \to \mathbb{F}.$$

The same proposition implies that $\hat{V}_d$ can be computed in time $n \cdot O(m' \cdot |\mathbb{H}|^2)$.

The following lemma describes a HIP that is used as a subroutine in the protocol. Its goal is to (interactively) ensure the consistency of two consecutive computed layers of a circuit.

Lemma 5.2 (GKR consistency protocol, [GKR15])**.** *Fix finite fields $\mathbb{H} \subseteq \mathbb{F}$, a low degree extension $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}$ and a layered arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}$ of fan-in 2, size s and depth d.*[23] *Assume that the adjacency predicate of C can be computed by a $\mathsf{polylog}(n)$-size formula ϕ_{adj} that can be constructed in $\mathsf{polylog}(n)$-time. Then, given a layer index $i \in [d]$ and a claim (r, v) about the i^{th} layer V_i, there exists a HIP that runs on explicit input (r, v) and holographic input V_{i+1} such that:*

- **Completeness.** *If $\hat{V}_i[r] = v$ and the prover honestly follows the protocol, then the verifier outputs a correct claim (r', v') about the holographic input: $\hat{V}_{i+1}[r'] = v'$.*
- **Soundness.** *If $\hat{V}_i[r] \neq v$, then for any (unbounded) cheating prover, w.p. at least $1 - \frac{m|\mathbb{H}| \cdot |\phi_{adj}|}{|\mathbb{F}|}$ over the verifier's coins, either the verifier rejects or it outputs (r', v') s.t. $\hat{V}_{i+1}[r'] \neq v'$.*
- **Complexity.** *The prover runs in time $\mathsf{poly}(|\mathbb{F}|^m, \log n)$ and the verifier runs in time $|\mathbb{H}| \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(m, \log(|\mathbb{F}|))$. The communication complexity is $m \cdot |\mathbb{H}| \cdot \mathsf{polylog}(n) \cdot \log(|\mathbb{F}|)$ and the number of rounds is $m + 1.5$.*

We proceed with the full description of the interactive proof of Theorem 5.1. The input to the protocol is a string $x \in \mathbb{F}^n$ and a layered arithmetic circuit $C = C_n : \mathbb{F}^n \to \mathbb{F}$ of depth $d = d(n)$, whose adjacency predicate can be computed by a $\mathsf{polylog}(n)$-size formula that can be constructed in $\mathsf{polylog}(n)$-time.

In what follows, for a string V, we use the notation $\hat{V} = \mathsf{LDE}_{\mathbb{F},\mathbb{H}}(V)$. For every $1 \leq i \leq d-1$, we take $\ell + 1$ to be the number of layers in the Merkle Tree (see Construction 3.3) whose leaves are $\hat{V}_i$, with respect to a UOWHFs family $\mathcal{H}$. Since the size of each layer V_i is fixed (to s), the number of functions required to build these trees is fixed to ℓ.

[23] We take $m = O(\log_{|\mathbb{H}|}(s))$, and this holds for the input layer as well, since $m' \leq m$.

The Flat-GKR protocol for $\mathcal{L}$ $(\mathcal{P}_{\mathsf{Flat}}, \mathcal{V}_{\mathsf{Flat}})$

1. $\mathcal{V}_{\mathsf{Flat}}$ uses the generating algorithm G from Theorem 3.2 to sample ℓ UOWHFs $h_1, ..., h_\ell \in_R \mathcal{H}$, and sends their description h to $\mathcal{P}_{\mathsf{Flat}}$.

2. $\mathcal{P}_{\mathsf{Flat}}$ commits to the computed layers of the circuit: it computes $C(x)$ and the hash roots $(y_i)_{i\in[d-1]}$ for every layer $\hat{V}_i$ of $C(x)$, with respect to h. It sends the hash roots to $\mathcal{V}_{\mathsf{Flat}}$, together with the path $p' = (p'_1, ..., p'_{\ell+1})$.[a]

3. $\mathcal{V}_{\mathsf{Flat}}$ checks that p' is a valid path with respect to y_1, and that $p'_{\ell+1} = 10^{n_{in}-1}$.[b] Otherwise, it immediately rejects.

4. For $i = 1, ..., d-1$, do the following in parallel:

   (a) Run the HIP $(\mathcal{P}_{\mathsf{HHR}}, \mathcal{V}_{\mathsf{HHR}})$ from Theorem 4.2, on explicit input (y_i, h) and holographic input V_i. If $\mathcal{V}_{\mathsf{HHR}}$ rejects then $\mathcal{V}_{\mathsf{Flat}}$ immediately rejects. Otherwise, the protocol ends with a claim (r, v) about $\hat{V}_i$.

   (b) Run the HIP from Lemma 5.2 on input claim (r, v) with respect to layers $\hat{V}_i$ and $\hat{V}_{i+1}$. If the verifier does not reject, denote its output claim, about $\hat{V}_{i+1}$, by (r', v').

   (c) If $i < d-1$:

      i. $\mathcal{P}_{\mathsf{Flat}}$ opens the commitment for (r', v'): it computes the leaf index $q' = \left\lceil \frac{r'}{n_{in}} \right\rceil$, and sends q' and the path $p = (p_1, ..., p_{\ell+1})$.

      ii. $\mathcal{V}_{\mathsf{Flat}}$ checks that p is a valid path with respect to y_{i+1} and q' and that it is consistent with v',[c] and rejects otherwise.

   Otherwise (i.e., $i = d-1$):

   - In the *holographic* case, the protocol outputs (r', v').
   - In the *non-holographic* case, the verifier checks if $\hat{V}_d[r'] = \mathsf{LDE}(x)[r'] = v'$. If the test fails, it rejects and otherwise it accepts.

[a] This path should be an opening for index 0 of the first (output) layer $\hat{V}_1$.

[b] This condition means that $C(x) = 1$, since $V_1 = C(x)0^{n_{in}-1}$, and the first s coordinates of $\hat{V}_1$ are V_1, as LDE is a systematic code.

[c] Formally, the verifier checks that $p_{\ell+1}[r''] = v'$ where $r'' = r' \pmod{n_{in}}$, i.e., r'' is the relative index of r' in the q'^{th} leaf.

5.1 Proof of Theorem 5.1

Fix an ensemble of circuits $(C_n)_{n\in\mathbb{N}}$ whose adjacency predicates can be computed by a $\mathsf{polylog}(n)$-size formula that can be constructed in $\mathsf{polylog}(n)$-time. For an input x of length n, denote the depth of $C = C_n$ by $d = d(n)$ and its size by $s = s(n)$.

Completeness. The perfect completeness follows from the protocol description, as well as from the perfect completeness of $(\mathcal{P}_{\mathsf{HHR}}, \mathcal{V}_{\mathsf{HHR}})$, proved in Theorem 4.2, and the perfect completeness of the HIP from Lemma 5.2.

Soundness. For the soundness condition, fix an input $x \notin \mathcal{L}$, and assume that there exists a cheating prover $\mathcal{P}^*_{\mathsf{Flat}}$ that runs in polynomial time, such that after interacting with it, the probability that the verifier $\mathcal{V}_{\mathsf{Flat}}$ accepts is ξ. That is,

$$\Pr[\mathcal{V}_{\mathsf{Flat}} \text{ accepts}] = \xi.$$

Let A denote this event.

Recall that Step (4) consists of $d-1$ parallel phases. Each phase consists of two HIPs and a test. For every $i \in [d-1]$, we use the notation below for the following events:

- Y_i: the hash root y_i that the prover sends in Step (2) is the correct hash root for $\hat{V}_i$ (that is, the encoding of the i^{th} layer of the circuit computed on input x) with respect to h;
- H_i: the HHR verifier does not reject in Step (4a), and the protocol's output claim (r, v) is correct;
- E_i: the verifier of the HIP from Lemma 5.2 does not reject in Step (4b), and the protocol's output claim (r', v') is correct;
- P_i: for $i > 1$, P_i is the event that p, the opening for layer i, is a valid path that passes the test of Step (4(c)ii). Notice that P_i concerns the opening sent in the $(i-1)^{\text{st}}$ phase. For $i = 1$, P_1 implies that p', the opening for layer 1, is a valid path that passes the test of Step (3).

Since $A \subseteq P_1$,

$$\xi = \Pr[A] = \Pr[A \wedge P_1] \leq \Pr[Y_1 \wedge P_1] + \Pr[A \wedge \neg Y_1]. \tag{8}$$

P_1 implies that p' is a valid path (see Definition 3.4) and $p'_{\ell+1} = 10^{n_{in}-1}$. Since $x \notin \mathcal{L}$, it holds that $C(x) = 0$, so $V_1 \neq 10^{n_{in}-1}$. If we also assume Y_1, namely that y_1 is the correct hash root with respect to $\hat{V}_1$, then the prover breaks the second preimage security of the commitment scheme as stated in Claim 3.6. We stress that the first preimage, that is, $\hat{V}_1$, is fixed *before* the UOWHFs are chosen, thus the security property indeed applies and

$$\Pr[Y_1 \wedge P_1] = negl(n_{in}).$$

We proceed with the event $(A \wedge \neg Y_1)$ and split into two cases.

- **Case 1:** Y_{d-1} is violated, i.e., y_{d-1} is a false hash root. Notice that $A \subseteq E_{d-1}$, since if the claim about the output layer is false, then the verifier rejects (or outputs a false claim, in the holographic case). Therefore, either H_{d-1} holds, and then the soundness guarantee of the HHR protocol is violated (because if the protocol's input is a false hash root, it should output a false claim), or $\neg H_{d-1}$ holds, and then $(\neg H_{d-1} \wedge E_{d-1})$ happens, which breaks the soundness condition of Lemma 5.2 (because if the protocol's input is a false claim, it should also output a false claim). Namely,

$$\begin{aligned}\Pr[A \wedge \neg Y_1 \wedge \neg Y_{d-1}] &\leq \Pr[\neg Y_{d-1} \wedge H_{d-1}] + \Pr[\neg H_{d-1} \wedge E_{d-1}] \\ &\leq n^{O(\delta)}/|\mathbb{F}| + (m \cdot |\mathbb{H}| \cdot |\phi_{adj}|)/|\mathbb{F}|,\end{aligned}$$

where the last inequality follows from Theorem 4.2 and Lemma 5.2, where $|\phi_{adj}| = \mathsf{polylog}(n)$.[24] The full details of these two reductions can be found in Claims 5.5 and 5.6 below, when setting $i^* = d-1$ and defining $D_{d-1} = (A \wedge \neg Y_{d-1})$. There, we formally show how to reduce $\mathcal{P}^*_{\mathsf{Flat}}$ to cheating strategies that break the soundness guarantee of the HIPs.

- **Case 2:** Y_{d-1} holds, i.e., y_{d-1} is a correct hash root. This means that the first layer has a false hash root (due to $\neg Y_1$) while the $(d-1)^{\text{st}}$ layer has a correct hash root (due to Y_{d-1}). Hence, by a hybrid argument, there exists a layer $i^* \in [d-2]$ such that y_{i^*} *is not* the correct hash root for $\hat{V}_{i^*}$ but y_{i^*+1} *is* the correct hash root for $\hat{V}_{i^*+1}$. That is,

$$\Pr\left[A \wedge \neg Y_1 \wedge Y_{d-1}\right] \leq \Pr\left[\exists i^* \in [d-2] \text{ s.t. } A \wedge \neg Y_{i^*} \wedge Y_{i^*+1}\right].$$

  The rest of the proof is dedicated to bounding the probability of this case.

We let D_i denote the event that $i \in [d-2]$ is the minimal index where $(A \wedge \neg Y_i \wedge Y_{i+1})$ holds. Observe that

$$\Pr\left[\exists i^* \in [d-2] \text{ s.t. } A \wedge \neg Y_{i^*} \wedge Y_{i^*+1}\right] = \Pr\left[\exists i^* \in [d-2] \text{ s.t. } D_{i^*}\right]. \tag{9}$$

Taking the *minimal* index in the definition of D_i is immaterial; any choice of a unique index i (e.g., the maximal) where $(\neg Y_i \wedge Y_{i+1})$ holds would work. The idea is to avoid paying for a Union bound over *all* layers in Proposition 5.4 (where in Proposition 5.3 it is unavoidable).

We split the event on the right-hand side of Equation (9) into two disjoint events, and the following two propositions bound the probability of each of them.

Proposition 5.3. $\Pr\left[\exists i^* \in [d-2] \text{ s.t. } D_{i^*} \wedge \neg E_{i^*}\right] \leq (d-2) \cdot negl(n_{in})$.

Proof. The stated event means that there exists a (minimal) layer i^* where the prover sends a false root for layer i^* and a correct root for the next layer (i^*+1), but the prover fails in cheating in either of the HIPs of phase i^* (Step (4a) and Step (4b) for $i = i^*$), and so (r', v') is a false claim about layer (i^*+1). By a Union Bound,

$$\Pr[\exists i^* \in [d-2] \text{ s.t. } D_{i^*} \wedge \neg E_{i^*}] \leq \sum_{i=1}^{d-2} \Pr[A \wedge Y_{i+1} \wedge \neg E_i] \leq \sum_{i=1}^{d-2} \Pr[P_{i+1} \wedge Y_{i+1} \wedge \neg E_i],$$

where the last inequality follows from the fact that $A \subseteq P_{i+1}$, due to the test performed in Step (4(c)ii).

Recall that the event $(P_{i+1} \wedge Y_{i+1} \wedge \neg E_i)$ implies that y_{i+1} is a correct commitment that the prover manages to open to a false value in a valid way. Applying the same arguments used to bound the probability of $(Y_1 \wedge P_1)$, together with Claim 3.6 (which is applicable, since $\forall i,\ |\hat{V}_i| = \mathsf{poly}(n) = \mathsf{poly}(n_{in})$), we conclude that each of these events happens with a negligible probability, namely:

$$\forall i \in [d-2],\ \Pr[P_{i+1} \wedge Y_{i+1} \wedge \neg E_i] = negl(n_{in}),$$

and the proposition follows. □

[24] In fact, the *degree* of the adjacency predicate is the one included in the soundness error. In the lemma's statement we trivially bound the degree with the size of the formula computing the predicate, which is promised to be polylogarithmic.

Proposition 5.4. $\Pr\left[\exists i^* \in [d-2] \ s.t. \ D_{i^*} \wedge E_{i^*}\right] \leq n^{O(\delta)}/|\mathbb{F}| + \left(m \cdot |\mathbb{H}| \cdot |\phi_{adj}|\right)/|\mathbb{F}|$.

Proof. The stated event means that in the first layer i^* where the prover sends a false root for layer i^* and a correct root for the next layer (i^*+1), it succeeds in cheating in one of the HIPs of phase i^*, which means that (r', v') is a correct claim about layer (i^*+1) although y_{i^*} is false. Since for any events X, Y it holds that $\Pr[X \wedge Y] \leq \Pr[X \mid Y]$, it suffices to bound

$$\Pr\left[E_{i^*} \mid \exists i^* \text{ s.t. } D_{i^*}\right].$$

Observe that if there exists i^* such that D_{i^*} holds, then it is unique, therefore i^* and hence the event E_{i^*} are well defined under the conditioning. The last probability is bounded by

$$\Pr\left[H_{i^*} \mid \exists i^* \text{ s.t. } D_{i^*}\right] + \Pr\left[\neg H_{i^*} \wedge E_{i^*} \mid \exists i^* \text{ s.t. } D_{i^*}\right],$$

i.e., if the root y_{i^*} is false, but the output claim about layer (i^*+1) is correct, then the prover manages to cheat in at least one of the HIPs. The next two claims bound the two terms above.

Claim 5.5. $\Pr\left[H_{i^*} \mid \exists i^* \ s.t. \ D_{i^*}\right] \leq n^{O(\delta)}/|\mathbb{F}|$.

Proof. We reduce $\mathcal{P}^*_{\mathsf{Flat}}$ to a cheating prover strategy $\mathcal{P}^*_{\mathsf{HHR}}$ that works as follows:

- Sample $h \leftarrow h_1, ..., h_\ell$ uniformly at random from $\mathcal{H}$, and send h to $\mathcal{P}^*_{\mathsf{Flat}}$.
- Receive $(y_i)_{i\in[d-1]}$ from $\mathcal{P}^*_{\mathsf{Flat}}$ and find i^*.
- Interact with $\mathcal{V}_{\mathsf{HHR}}$ on explicit input (y_{i^*}, h) and holographic input V_{i^*}. Feed $\mathcal{P}^*_{\mathsf{Flat}}$ with $\mathcal{V}_{\mathsf{HHR}}$'s messages, and vice versa. At the end of the interaction, $\mathcal{V}_{\mathsf{HHR}}$ either rejects or outputs a claim (r, v) about $\hat{V}_{i^*}$.

Since D_{i^*} implies $\neg Y_{i^*}$, then $(y_{i^*}, h) \notin \mathcal{L}_{\mathsf{HHR}}$. Moreover, H_{i^*} implies that $\mathcal{V}_{\mathsf{Flat}}$, that executes $\mathcal{V}_{\mathsf{HHR}}$, does not reject. By the soundness guarantee of the HHR protocol, this means that the probability (over $\mathcal{V}_{\mathsf{HHR}}$'s coin tosses) that (r, v) is correct (namely, that H_{i^*} holds) is at most $n^{O(\delta)}/|\mathbb{F}|$. The claim follows by reducing the randomized strategy $\mathcal{P}^*_{\mathsf{HHR}}$ to an (unbounded) deterministic strategy by fixing its best choice of randomness. □

Claim 5.6. $\Pr\left[\neg H_{i^*} \wedge E_{i^*} \mid \exists i^* \ s.t. \ D_{i^*}\right] \leq \left(m \cdot |\mathbb{H}| \cdot |\phi_{adj}|\right)/|\mathbb{F}|$.

Proof. The proof follows along similar lines to Claim 5.5. Denote the HIP from Lemma 5.2 by $(\mathcal{P}, \mathcal{V})$. We reduce $\mathcal{P}^*_{\mathsf{Flat}}$ to a cheating prover strategy $\mathcal{P}^*$ that works as follows:

- Sample $h \leftarrow h_1, ..., h_\ell$ uniformly at random from $\mathcal{H}$, and send h to $\mathcal{P}^*_{\mathsf{Flat}}$.
- Receive $(y_i)_{i\in[d-1]}$ from $\mathcal{P}^*_{\mathsf{Flat}}$ and find i^*.
- Simulate $\mathcal{V}_{\mathsf{HHR}}$ on explicit input (y_{i^*}, h) and holographic input V_{i^*}. At the end of the interaction, $\mathcal{V}_{\mathsf{HHR}}$ either rejects or outputs a claim (r, v) about $\hat{V}_{i^*}$.
- Interact with $\mathcal{V}$ on explicit input (r, v) and holographic input V_{i^*+1}. Feed $\mathcal{P}^*_{\mathsf{Flat}}$ with $\mathcal{V}$'s messages, and vice versa. At the end of the interaction, $\mathcal{V}$ either rejects or outputs a claim (r', v') about $\hat{V}_{i^*+1}$.

Once again, if E_{i^*} happens, then the verifiers do not reject. Notice that $\neg H_{i^*}$ implies that (r, v) is a false claim about $\hat{V}_{i^*}$. On the other hand, E_{i^*} implies that (r', v') is a correct claim about $\hat{V}_{i^*+1}$. By the soundness guarantee of Lemma 5.2, this means that the probability of this event (over $\mathcal{V}$'s coin tosses) is at most $(m \cdot |\mathbb{H}| \cdot |\phi_{adj}|)/|\mathbb{F}|$. Again, the claim follows by reducing the randomized strategy $\mathcal{P}^*$ to a deterministic one. □

This concludes the proof of Proposition 5.4. □

All in all, getting back to Equation (8), by the choice of $\mathbb{F}$ and $\mathbb{H}$ we conclude that

$$\xi \leq (d-1) \cdot negl(n_{in}) + 2\left(n^{O(\delta)}/|\mathbb{F}| + (|\mathbb{H}| \cdot \mathsf{polylog}(n))/|\mathbb{F}|\right) = n^{O(\delta)}/|\mathbb{F}|.$$

Complexity. The stated complexity follows from the complexity of the HIPs, that are run on holographic inputs of size $s(n) = \mathsf{poly}(n)$. Recall that $d = O(\log n) \cdot D(n)$.

Theorem 3.2 promises that the verifier can sample the UOWHFs in time $\mathsf{poly}(\kappa) = n^{O(\delta)}$. Theorem 4.2 implies that the verifier $\mathcal{V}_{\mathsf{HHR}}$ runs in time $n^{O(\delta)}$. In Step (4b), it runs in time $\tilde{O}(n^{\delta})$, due to Lemma 5.2. Checking that a path is valid takes $\ell \cdot \mathsf{poly}(\kappa) = \mathsf{polylog}(n) \cdot n^{O(\delta)} = n^{O(\delta)}$ time. Overall, we get that:

$$\mathcal{V}\mathsf{time} \leq n^{O(\delta)} + O(\log n) \cdot D(n) \cdot \left(n^{O(\delta)} + \tilde{O}(n^{\delta}) + n^{O(\delta)}\right) = D(n) \cdot n^{O(\delta)}.$$

For the non-holographic case, in Step (4c) the verifier runs in time $n \cdot O(|\mathbb{H}|^2) = n^{1+2\delta}$, which yields an overall running time of $D(n) \cdot n^{O(\delta)} + n^{1+2\delta} = (D(n) + n) \cdot n^{O(\delta)}$.

The analysis of the communication complexity follows from the same arguments, and we get that it is bounded by $D(n) \cdot n^{O(\delta)}$ as well.

Notice that the number of rounds in Lemma 5.2 is $m + 1.5$, where $m = \log_{n^{\delta}}(\mathsf{poly}(n)) = O(1/\delta)$. Hence, the number of rounds is dominated by the round complexity of the HHR protocol, which is $O(1/\delta^3)$ rounds.

Acknowledgments

We thank Oded Goldreich for helpful and illuminating conversations and comments. Noga Amit would like to dedicate this paper to her dearest Nimrod Pansky, who was with her in spirit throughout this work, and will remain in her heart forever.

References

[AM13] Benny Applebaum and Yoni Moses. Locally computable UOWHF with linear shrinkage. *IACR Cryptol. ePrint Arch.*, page 423, 2013.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 37–56, 1988.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 671–684. ACM, 2018.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making uowhfs practical. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 1997.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for P from LWE. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 68–79. IEEE, 2021.

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[Dam89] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[FL91] Lance Fortnow and Carsten Lund. Interactive proof systems and alternating time-space complexity. In Christian Choffrut and Matthias Jantzen, editors, *STACS 91, 8th Annual Symposium on Theoretical Aspects of Computer Science, Hamburg, Germany, February 14-16, 1991, Proceedings*, volume 480 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 1991.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27, 2015.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol17] Oded Goldreich. On the doubly-efficient interactive proof systems of GKR. *Electron. Colloquium Comput. Complex.*, TR17-101, 2017.

[GR17] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 39:1–39:43, 2017.

[GR20] Oded Goldreich and Guy N. Rothblum. Constant-round interactive proof systems for AC0[2] and NC1. In Oded Goldreich, editor, *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 326–351. Springer, 2020.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. *Adv. Comput. Res.*, 5:73–90, 1989.

[GVW02] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

[HHRS15] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 850–858. IEEE Computer Society, 2018.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989.

[Ish20a] Yuval Ishai. Zero-knowledge proofs from information-theoretic proof systems - part i. Available at https://zkproof.org/2020/08/12/information-theoretic-proof-systems/, 2020.

[Ish20b] Yuval Ishai. Zero-knowledge proofs from information-theoretic proof systems - part ii. Available at https://zkproof.org/2020/10/15/information-theoretic-proof-systems-part-ii/, 2020.

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1987.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions, 2005. jkatz@cs.umd.edu 13051 received 16 Sep 2005, last revised 24 Sep 2005.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 162–194. Springer, 2018.

[KRR22] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: The power of no-signaling proofs. *J. ACM*, 69(1):1:1–1:82, 2022.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Mer89] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *FOCS*, pages 436–453, 1994.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 33–43. ACM, 1989.

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41:333–338, 1987.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394. ACM, 1990.

[RRR16] 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.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient batch verification for UP. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 22:1–22:23, 2018.

[RV22] Ron D. Rothblum and Prashant Nalini Vasudevan. Collision-resistance from multi-collision-resistance. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 503–529. Springer, 2022.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *STOC*, pages 793–802, 2013.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987.

[Sud95] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*, volume 1001 of *Lecture Notes in Computer Science*. Springer, 1995.

[Tha22] Justin Thaler. Proofs, arguments, and zero-knowledge. Available at https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, 2022.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2005.

A IPs, HIPs and HIPPs: Protocols and Assumptions

We review results on interactive proof systems, with an eye towards the relevant parameters for our work. We begin in Section A.1 by elaborating on plausible assumptions about bounds on the power of IPs. We give overviews of HIP and HIPP constructions in Sections A.2 and A.3 (respectively). Finally, in Section A.4, we present the full proof of Claim 3.22, deferred from the preliminaries section due to its length.

A.1 Assumptions About The Power of IPs

As discussed in Remark 1.2, we can significantly relax the notion of an interactive proof if we only require an infinitesimal gap between the completeness and the soundness error (in particular, the gap can be smaller than the inverse of the computation size).We briefly describe how known results become straightforward under this relaxation:

1. For a (sufficiently uniform) size-S depth-D circuit, we can get a protocol with $O(D)$ rounds and communication: each round begins with a claim about the alleged value of a gate in the circuit. The prover sends the alleged values of the gate's children, and the verifier chooses to recurse on one of them at random. The protocol has perfect completeness. In each round, if we began with a false claim, then with probability at least $(1/2)$ the new claim will also be false. Thus, the soundness error is $(1 - 2^{-D})$.

   The above can be viewed as a "weak sauce" or substandard version of the GKR protocol: both protocols peel back the circuit computation in a layer-by-layer manner, but the GKR protocol achieves this "peeling" with robust soundness, and the end protocol has a constant gap between the completeness and the soundness error.

2. For a T-time S-space computation, and for any desired $\sigma \in (0, 1)$, we can get a protocol with $O(T^\sigma \cdot S)$ communication and $O(1/\sigma)$ rounds. In each round we begin with a claim about a T'-time computation. The prover sends $k = T^\sigma$ intermediate computation states, and the verifier recurses on a random pair of subsequent states, where the claim in the next round is that there is a computation path of length $T'' = T'/T^\sigma$ between them. The protocol has perfect completeness by construction. If we began a round with a false claim, then at least one pair of subsequent states sent by the prover do *not* have a computation path of length T'' between them, and the verifier will catch this with probability $1/k$. Thus, the protocol's soundness error is $(1 - 1/T)$.

   The above can be viewed as a "weak sauce" or substandard version of the RRR protocol: both protocols break the computation into k subclaims about shorter computations and recurse, but the RRR protocol uses a robust batch-verification technique to achieve a constant gap between the completeness and the soundness error.

Our goal above was demonstrating that it is much easier to construct IPs with an infinitesimal completeness-soundness gap. Indeed, the celebrated achievements of the IPs literature, starting with the $\mathsf{IP} = \mathsf{PSPACE}$ protocol [LFKN92, Sha92], have revealed that (miraculously) in many cases IPs with a large gap can achieve much of the expressive power of the relaxed notion.

This brings us to our main point regarding limits on the power of constant-round IPs. Namely, for the class of languages computable by log-space uniform linear-depth poly-size circuits, *it is*

not known how to construct a constant-round DEIP with **any** *gap between the completeness and the soundness error.* We find it plausible to conjecture that no such proof system exists: both for infinitesimal-gap DEIPs, and certainly for constant-gap DEIPs. Indeed, even for AC^1, it is completely unclear how to construct a constant-round DEIP with any gap.

IPs and alternating computation. We conclude this section by remarking that the relaxed infinitesimal-gap IP notion is closely related to the notion of *alternating computation*. An alternating Turing Machine [CKS81] is an extension of non-deterministic computation, where the machine can make both existential and universal moves. There is a close relationship between ATMs and IPs: Goldwasser and Sipser [GS89] showed that IPs are equivalent to ATMs that alternate between existential and probabilistic moves (see also the work of Fortnow and Lund [FL91]). In particular, any language that has an r-round, c-communication doubly-efficient IP with verifier runtime $\mathcal{V}\mathsf{time}$ is also in the class $\mathsf{ATIME}(r, c, \mathcal{V}\mathsf{time})$ of languages decidable by an alternating Turing machine with r quantifier alternations between existential and universal moves, where the total number of such moves is c, and the machine can also make $\mathcal{V}\mathsf{time}$ many probabilistic computation steps (if the protocol has perfect completeness, then $\mathcal{V}\mathsf{time}$ many *deterministic* computation steps are sufficient). In the reverse direction, alternating TMs are equivalent to the relaxed notion of IPs with *any* gap between completeness and soundness (the notion discussed above).

Thus, under the plausible assumption that the class of languages computable by log-space uniform linear-depth circuits is not in $\mathsf{ATIME}(r, O(n^{1+\sigma}), O(n^{1+\sigma}))$ for any constants r, σ, there are no constant-round doubly-efficient interactive proofs for this class.

A.2 Constant Round HIP for $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$

We review the higher-level structure of the GR construction [GR20], with an eye towards the (minor) changes needed for the proof of Theorem 3.25. First, their theorem was stated for Boolean $\mathsf{AC}^0[\oplus]$ circuits, but it can be separated / abstracted into two steps: (i) transforming a highly uniform $\mathsf{AC}^0[\oplus]$ circuit to an arithmetic circuit over $\mathbb{GF}[2]$ whose multiplication gates have bounded fan-in and an incidence function. This is the transformation we refer to in Theorem 3.18. Next, (ii) they construct a proof system for $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuits under appropriate conditions on the uniformity and the fan-in of multiplication gates. In this section, we focus on this second step. Moreover, the GR protocol was not holographic, but it can be made holographic in a straightforward manner.

W.l.o.g, we assume that the $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit C is layered, and that all gates in each layer are either of the addition type or the multiplication type. The protocol unwinds the computation layer by layer in sequential iterations, where in each iteration the prover and the verifier execute a constant-round layer-consistency sub-protocol. The i-th execution of the sub-protocol begins with a claim $w^{(i)}$ about value of the LDE of layer i at a location $r^{(i)}$, and ends with a claim $(r^{(i+1)}, w^{(i+1)})$ about the value of the LDE of layer $(i+1)$. As usual, if the input claim is true then the output claim will also be true. If the input claim is false, then w.h.p. over the verifier's coins the output claim will also be false. The main step in this sub-protocol is an interactive sum-check, but the details vary between layers of addition gates and layers of multiplication gates.

Addition layers. For addition layer i, suppose that both layer i and layer $(i+1)$ are of width $|\mathbb{H}|^m$, and that the gates in both layers are labeled by vectors in $\mathbb{H}^m$. We use the (arithmetization of the) adjacency predicate $\widehat{\phi}_{adj}(j,k)$ that outputs 1 if the addition gate j in layer i is fed by gate k in layer $(i+1)$. We remark that the adjacency predicate should take bit strings as its input,

whereas here we feed it with vectors of field elements, but one can translate the field elements into the appropriate strings using an easy-to-compute translation function of degree $|\mathbb{H}|$ (for more details, see Claim 3.22).

Let $V^{(i)}$ be the vector of values of the gates in layer i, and $\widehat{V}^{(i)}$ its low-degree extension w.r.t. $\mathbb{H}, \mathbb{F}$ (see Section 3.2). Similarly, let $V^{(i+1)}$ be the vector of values of the gates in layer $(i+1)$ and $\widehat{V}^{(i+1)}$ its low-degree extension. Finally, let $\widehat{\tau}(x,z) = \widehat{\tau}_x(z)$ be the functions used to define the low-degree extension, see Section 3.2 and Proposition 3.10.

The starting claim is that $\widehat{V}^{(i)}[r^{(i)}] = w^{(i)}$. Decomposing the claim about the LDE of the i-th layer into a sum, this claim can be verified via a sum-check protocol:

$$\begin{aligned}
\widehat{V}^{(i)}[r^{(i)}] &= \sum_{j \in \mathbb{H}^m} \widehat{\tau}(j, r^{(i)}) \cdot V^{(i)}[j] \\
&= \sum_{j \in \mathbb{H}^m} \widehat{\tau}(j, r^{(i)}) \cdot \left(\sum_{k \in \mathbb{H}^m} \widehat{\phi}_{adj}(j,k) \cdot V^{(i+1)}[k] \right) \\
&= \sum_{j \in \mathbb{H}^m, k \in \mathbb{H}^m} \widehat{\tau}(j, r^{(i)}) \cdot \widehat{\phi}_{adj}(j,k) \cdot \widehat{V}^{(i+1)}[k].
\end{aligned}$$

The sum-check protocol outputs vectors $r', r'' \in \mathbb{H}^m$ and claims about the values of $\widehat{\tau}(r', r^{(i)})$, $\widehat{\phi}_{adj}(r', r'')$ and $\widehat{V}^{(i+1)}[r'']$. The first two claims can be verified by the verifier, the last claim is the output claim for this layer's subprotocol (i.e. $r^{(i+1)} = r''$).

In terms of the degree of the summands, the degree in j and in k is as follows. Notice that, as opposed to other usages (e.g., Lemma 3.8) where we consider the *total* degree (that is, the sum of the degrees in each of the variables), here we consider the *individual* degree, since this is the one that appears in the soundness error of the sum-check protocol.[25]

$$\begin{aligned}
\mathit{degree} &= O\left(|\mathbb{H}| + \mathit{deg}(\widehat{\phi}_{adj}) \right) \\
&= O\left(n^{\delta} + \mathit{deg}(\widehat{\phi}_{adj}) \right).
\end{aligned} \tag{10}$$

The communication complexity and verifier runtime in this subprotocol are thus $\tilde{O}(\mathit{degree})$, and the soundness error is $O(m \cdot \frac{\mathit{degree}}{|\mathbb{F}|}) = O(\mathit{degree}/|\mathbb{F}|)$. If $\mathit{deg}(\widehat{\phi}_{adj}) = d$, as assumed in the theorem's statement, we get that the soundness error is $O(n^{\delta} + d)/|\mathbb{F}|$.

Multiplication layers. For multiplication layers, we proceed as above, but now we have (an arithmetication of) the incidence function $\widehat{\phi}_{incd}(j,k)$ that outputs the label of the k-th gate in layer $(i+1)$ that feeds into gate j in layer i. Let $m' = \log_{|\mathbb{H}|}(f_{in}) = O(1)$ be the dimension of the LDE

[25] In more detail, recall that in each round of the sum-check protocol, the prover sends a *univariate* polynomial. This means that the soundness error in each round is proportional to the individual degree, and that the overall soundness error of the sum-check execution is the sum (over all rounds) of these errors.

of the "index" k. We again decompose the claim about layer i's LDE:

$$\begin{aligned}\widehat{V}^{(i)}[r^{(i)}] &= \sum_{j\in\mathbb{H}^m} \widehat{\tau}(j, r^{(i)}) \cdot V^{(i)}[j] \\ &= \sum_{j\in\mathbb{H}^m} \widehat{\tau}(j, r^{(i)}) \cdot \left(\prod_{k\in\mathbb{H}^{m'}} V^{(i+1)}[\widehat{\phi}_{incd}(j,k)] \right) \\ &= \sum_{j\in\mathbb{H}^m} \left(\prod_{k\in\mathbb{H}^{m'}} \widehat{\tau}(j, r^{(i)}) \cdot \widehat{V}^{(i+1)}[\widehat{\phi}_{incd}(j,k)] \right).\end{aligned}$$

Here we have a sum of products. The (individual) degree of the "internal" function is $\left(|\mathbb{H}| \cdot deg(\widehat{\phi}_{incd})\right)$. The total number of multiplicants in the product is $f_{in} = O(n^{\delta})$. Thus, the degree in each round of the sum-check protocol (actually, a sum-product check a la [Sha92]) is bounded by:

$$degree = O\left(n^{2\delta} \cdot deg(\widehat{\phi}_{incd})\right). \tag{11}$$

The communication complexity and verifier runtime are again $\tilde{O}(degree)$, and the soundness error is $O(degree/|\mathbb{F}|)$ since $m = O(1)$. Once again, if $deg(\widehat{\phi}_{incd}) = d$, as assumed in the theorem's statement, we get that the soundness error is $O(n^{2\delta} \cdot d)/|\mathbb{F}| = n^{O(\delta)} \cdot d/|\mathbb{F}|$. Assuming C has size $n^{C'}$ and depth D', the number of rounds for executing each sum-check is $O(C'/\delta)$, and thus the overall round complexity is $O(d \cdot C'/\delta)$.

Remark A.1 (Multi-output circuits)**.** *The protocol we described also applies to cases where we want to verify the output of an arithmetic circuit that has multiple (say $\ell \geq 1$) output wires. By convention, we will verify that the value of all output wires is 1. This is done by simply picking the input claim to the first sub-protocol's execution to be a random location $r^{(1)}$ on the LDE of the output layer, where the claimed value $v^{(1)} = \mathsf{LDE}(1^{\ell})[r^{(1)}]$. The choice of 1^{ℓ} is arbitrary; any vector that is known to the verifier would work (however, it is crucial that the verifier knows this output vector in order to compute $v^{(1)}$).*

A.2.1 Setting the field size

In this work, defining $|\mathbb{F}| = \mathsf{poly}(n)$ without specifying the polynomial was sufficient for our needs. Indeed, taking a step back, the only condition that $|\mathbb{F}|$ had to satisfy is to be big enough such that the soundness error $n^{O(\delta)}/|\mathbb{F}|$ would be $o(1)$. Thus, as is, it is unclear that $|\mathbb{F}|$ can be taken to be *almost-linear* in $|\mathbb{H}|$, that is, $|\mathbb{F}| = |\mathbb{H}| \cdot n^{\alpha}$, for some $\alpha \ll \delta$. In this subsection, we show that for Flat-GKR (namely, for this work), it is possible.

We start with backtracking the $n^{O(\delta)}/|\mathbb{F}|$ soundness error of Flat-GKR (see Theorem 5.1), when we only consider errors that are (strictly) bigger than $O(|\mathbb{H}|/|\mathbb{F}|)$, because for smaller ones, taking $|\mathbb{F}|$ to be almost-linear in $|\mathbb{H}|$ is enough. This $n^{O(\delta)}/|\mathbb{F}|$ term is implied by the soundness error of HHR (see Theorem 4.2). There, it comes from the soundness error of the $(\mathcal{P}_i, \mathcal{V}_i)$ protocol. Taking a closer look at its soundness analysis (see Lemma 4.6), the $n^{O(\delta)}/|\mathbb{F}|$ term stems from the HIP of Theorem 3.25 (the theorem that is proved in this section), and from ρ, that is, the soundness error of the HIPP that instantiates Lemma 4.4. Since we use the HIPP of Theorem 3.26, we get that

ρ is the sum of $\lambda(n)$ and the HIP's soundness error,[26] where $\lambda(n)$ is smaller than any polynomial (see Proposition A.4). We conclude that we are only interested in the soundness error of the HIP of Theorem 3.25.

As mentioned earlier, this error originates in the degree of the multiplication gates (as it is bigger than the one of addition gates, see Equations (10) and (11), and the following arguments apply for addition gates as well). In particular, we "pay" with soundness error for this degree, due to the Schwartz-Zippel lemma (see Lemma 3.8), according to which we get an $O(degree/|\mathbb{F}|)$ error probability.

Let us review each of the multiplicants of Equation (11). Taking d to denote the degree of $\widehat{\phi}_{incd}$, we get that the overall degree of the entire sum is upper-bounded by:

$$degree = O(f_{in} \cdot n^{\delta} \cdot d). \tag{12}$$

Thus, if we prove that $f_{in} \leq n^{\beta}$ for some $\beta \ll \delta$, and that $d \leq n^{o(1)}$, we can take $\beta < \alpha \ll \delta$, and define $|\mathbb{F}| = n^{\delta+\alpha} = |\mathbb{H}| \cdot n^{\alpha}$. This way, we promise that the soundness error $n^{O(\delta)}/|\mathbb{F}|$ is $o(1)$.

First, we prove that $f_{in} \leq n^{\beta}$ for some $\beta \ll \delta$. Recall that for every gate in each of the circuits that are used throughout the work, it also holds that $f_{in} \leq n^{\delta+o(1)}$. As already done in Section 4.4.1 (see Footnote 19), we use the following iterative process: we scan the gates of the circuits from top to bottom (i.e., from the output layer to the input layer), and search for a gate u such that $f_{in}(u) > n^{\delta}$. Whenever we find this u, we create new n^{β} gates of the same type of u, such that each of them has $f_{in} \leq n^{\delta+o(1)}/n^{\beta}$. These are the new inputs to u. We repeat this process for any such u. This process halts when the condition is met, and it increases the size of the circuit by at most a polynomial factor, whereas its depth is increased by a multiplicative factor of $O(\delta/\beta)$.

Regarding the degree d, in each of the usages of the theorem, we proved that indeed in holds that $d \leq n^{o(1)}$ (in fact, we even proved something stronger: that $d \leq O(\log n)$).

Finally, for any given $\alpha = \delta/C$ where $C \in \mathbb{N}$ is some universal constant (i.e., independent of δ), by taking $\beta = \delta/(C+1)$, we may define $\mathbb{F}$ such that $|\mathbb{F}| = n^{\delta+\alpha}$ without hurting the soundness of the argument system. The depth of the circuits is still a constant independent of δ, as $O(\delta/\beta) = C+1$.

Remark A.2 (No $\mathsf{LDE}([\mathsf{LDE}(\cdot)]_2)$ in HHR)**.** *We remark that if the input x to the protocol is promised to be a binary representation of an LDE, i.e. $x = [\mathsf{LDE}(v)]_2$, then a naive application of the protocol yields a claim about $\mathsf{LDE}([\mathsf{LDE}(v)]_2)$. Indeed, this is the case in some of our usages of this theorem (namely, in implementing $(\mathcal{P}_{\ell}, \mathcal{V}_{\ell})$, see Section 4.3), as we always take the leaves of the tree to be in an LDE form, and we always convert them to binary (see Remark 3.7).*

We wish, instead, to get a claim directly about $\mathsf{LDE}(v)$. To do this, instead of running the protocol on the $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit C, we can run it on the circuit C' that takes as input v itself, computes $\mathsf{LDE}(v)$, then computes $[\mathsf{LDE}(v)]_2$, and then computes C on $[\mathsf{LDE}(v)]_2$. We can apply the theorem to C', as the LDE can be computed in $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ with multiplication gates of fan-in $|\mathbb{H}|$, and the binary representation of a field element can be computed in $O(\log n)$-highly uniform $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ with multiplication gates of fan-in $|\mathbb{F}|$ (see Claim 3.22). Recall that in the previous section, we proved that the size of $|\mathbb{F}|$ can be taken to be smaller than $n^{2\delta}$. This means that reducing the fan-in of C' from $|\mathbb{F}|$ to $|\mathbb{H}| = n^{\delta}$ can be done when only squaring the circuit's

[26] Recall that we only considered errors bigger than $O(|\mathbb{H}|/|\mathbb{F}|)$. Revisiting Equation (13), an exact calculation of the probability that at least one of the new forked inputs will have the (new) claimed distance from its (new) claims is $(|\mathbb{F}|-1)^{-1} + e^{-\mu/\log(|\mathbb{H}|)}$. By taking $\mu = O(n_{imp}^{o(1)} \cdot \log(|\mathbb{H}|))$, we get that this term is $o(1)$. For more details see the next section of the appendix, Section A.3, and [RVW13, Theorem 3.12, Theorem 3.8] with $r = O(1)$.

depth (and increasing its size by a polynomial factor). Thus, the circuit C' satisfies the conditions of Theorem 3.25.

A.3 Constant Round HIPP for $\mathsf{AC}^0[\oplus]$ and $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$

We give an overview for the proof of Theorem 3.26, an oblivious constant-round HIPP for (sufficiently uniform) triplet languages computable in $\mathsf{AC}^0[\oplus]$ or $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$. Our construction is based on the RVW IPP [RVW13], with a few twists and a careful analysis of that protocol's output claim about the implicit input. Our overview emphasizes the novel and important points for our analysis.

We briefly recall the setup and the goal of the protocol (see Definition 3.24 for the definition of HIPPs). The input is a triplet (x, y, z): an explicit input $x \in \{0,1\}^{n_{exp}}$, an implicit input $y \in \{0,1\}^{n_{imp}}$, and a holographic input $z \in \{0,1\}^{n_{hol}}$. The proximity parameter is $\varepsilon = \varepsilon(n_{imp})$. The IPP is *oblivious*: the verifier doesn't access the inputs (y, z). Rather, it interacts with the prover, accesses (only) the explicit input x, and in the end of the interaction it either rejects the prover's assertion, or outputs two claims about the input: (i) a claim on a subset Q of the coordinates of the implicit input y, where this claim is specified by a predicate ψ, and (ii) a holographic claim about z, asserting the value of a single location in z's low-degree extension.

Phase I: distance amplification. The protocol begins by running t parallel executions of a HIP protocol for the claimed circuit computation. Aiming for a constant number of rounds, we use the HIP of GR, as detailed in Theorem 3.25, where the explicit input to the HIP is empty and the holographic input is (x, y, z). Although x is explicit and the verifier holds its value, we put it into the holographic input to get a better dependence on n_{exp} in the verification time. The low-degree extension used is w.r.t the fields $\mathbb{H}$ and $\mathbb{F}$ (see Section 3.2), where the LDE of (x, y, z) is of dimension m_{total}, and the LDE of the implicit input y is of dimension m. If the HIP verifier rejects (in any execution), then we reject immediately. Otherwise, the HIP executions result in a vector $J' \in (\mathbb{F}^{m_{total}})^t$ of locations in the LDE of (x, y, z), and a vector $\vec{v}' \in \mathbb{F}^t$ of claims about $\mathsf{LDE}(x, y, z)[J'] = \vec{v}'$. By Proposition 3.12, we can partition the LDE claims into:

(i) claims about the explicit input, which will be merged into a single claim that the verifier checks explicitly, using the process described in Claim 3.13;

(ii) claims about the holographic input, which will also be merged into a single output holographic claim in the same manner, and

(iii) claims about a set $J \in (\mathbb{F}^m)^t$ of locations in the LDE of the implicit input y, asserting that they get values $\vec{v} \in \mathbb{F}^t$.

In the completeness case, all claims will be true. In the soundness case, w.h.p. either the verifier rejects in the HIP, or it rejects after it checks the claim about the explicit input, or the holographic claim is false, or the implicit input y *is at Hamming distance at least* $\tilde{\Omega}(t)$ from satisfying that $\mathsf{LDE}(y)[J] = \vec{v}$. The main goal in the protocol is reducing this claim to a claim about a subset of the coordinates of the implicit input (in the IPP context, this corresponds to achieving sublinear query complexity into the input).

Before moving to this second phase, we remark on the complexity of the first phase, which is incurred by running t repetitions of the HIP. Recall that $s(n)$ is the length of the advice string to the circuit and that the circuit's size is denoted by n^C. Looking ahead, we will take $t = \varepsilon \cdot n_{imp} \cdot n_{imp}^{o(1)}$, so we get that the communication complexity is $(\varepsilon \cdot n_{imp} \cdot n^{O(\delta)} \cdot s(n)^{1+o(1)})$, the verifier runs in time $n^{O(\delta)} \cdot \left(\varepsilon \cdot n_{imp} \cdot s(n)^{1+o(1)} + n_{exp}\right)$, the prover runs in time $\mathsf{poly}(n)$ and the number of rounds

is $O(C/\delta)$.

Phase II: iterated fork-and-reduce. The second phase reduces the claim $\mathsf{LDE}(y)[J] = \vec{v}$ to a *sublinear* claim about a subset Q of y's coordinates. If the initial claim is true, then the sublinear claim will also be true. On the other hand, if y is *far* from satisfying the input claim, then the sublinear claim will be false.

The protocol proceeds in rounds, where in each round we begin with a set of input-claim pairs (a set of inputs, and a set of claims about each input). We "fork" each input-claim pair into $f = \log(|\mathbb{H}|) = n_{imp}^{o(1)}$ new input-claim pairs, where the new inputs are of a reduced dimension. In more detail (and with careful indexing), there are r rounds, each indexed by $i \in [r]$. Each round begins with f^{i-1} input-claim pairs, indexed by $j \in [f]^{i-1}$. The i-th round begins with:

- A vector $(y^{(i,j)})_{j\in[f]^{i-1}}$ of f^{i-1} inputs, where each input is a column vector of length $|\mathbb{H}|^{m-i}$.

  The first round begins with a single input $y^{(0,(\cdot))} = y$, of length $\mathbb{H}^m$ (we take $(\cdot)$ to indicate a vector of dimension 0).

- A collection of t claims about each input, where the claims are all about the LDE of that input, and are specified by a t-dimensional vector $(J^{(i,j)})_{j\in[f]^{i-1}}$ of locations and a t-dimensional vector $(\vec{v}^{(i,j)})_{j\in[f]^{i-1}}$ of claimed values, where the claim is that:

  $$\forall j \in [f]^{i-1} : y^{(i,j)}[J^{(i,j)}] = \vec{v}^{(i,j)}.$$

  The first round begins with a single claim specified by $J^{(0,(\cdot))} = J$ and $\vec{v}^{(0,(\cdot))} = \vec{v}$.

- A distance bound $\delta^{(i,j)}$ on the fractional distance of $y^{(i,j)}$ from satisfying the claims. In the completeness case, all inputs satisfy their claims, whereas in the soundness case, w.h.p. in each round at least one $y^{(i,j)}$ is $\delta^{(i,j)}$-far from satisfying the claims.

  In the first round we have:

  $$\delta^{(0,(\cdot))} = \frac{\tilde{\Omega}(t)}{n_{imp}} = \varepsilon \cdot n_{imp}^{o(1)}$$

We proceed to describe the execution rounds, and how we fork each input into f new executions. In round i, we refer to the j-th "execution path", which operates on the (i,j)-th input-claim pair. As described in Section 3.2, the LDE of a vector $y' \in \mathbb{F}^{|\mathbb{H}|^{m'}}$ can be viewed as a matrix in the following way: We view y' itself as a matrix with dimensions $|\mathbb{H}| \times |\mathbb{H}|^{m'-1}$, thus we can refer to the "rows" of y', which are the rows of this matrix. The LDE of y' is a matrix with dimensions $|\mathbb{F}| \times |\mathbb{F}|^{m'-1}$. For $i \in [|\mathbb{H}|]$, the i-th row of the LDE is obtained by encoding the row y'_i of y' as an LDE. The j-th column of $\mathsf{LDE}(y')$ is the obtained by encoding the column vector $(y'_1[j], \ldots, y'_{|\mathbb{H}|}[j])^T$ (also as an LDE).

The i-th round. In the i-th round, for the (i,j)-th execution, the protocol begins with the prover sending $|\mathbb{H}|$ vectors of claimed values for the LDE of each row h of $y^{(i,j)}$. I.e., the prover sends $\{W_h^{(i,j)}\}_{h\in\mathbb{H}}$, and claims that:

$$\forall h \in \mathbb{H},\ \mathsf{LDE}(y_h^{(i,j)})[J^{(i+1,(j,k))}] = W_h^{(i,j)},$$

where $J^{(i+1,(j,k))}$ is derived from $J^{(i,j)}$ by simply erasing the first coordinate of each location.

Creating the f forks. The verifier sends f responses, where the k-th response generates the k-th fork of the (i,j)-th execution. In the k-th fork, the verifier picks a (random) subset of 2^k of the rows of the input $y^{(i,j)}$. It also picks a random coefficient in $\mathbb{F}$ for each of these rows. Let $A^{(i+1,(j,k))} \in \mathbb{F}^{|\mathbb{H}|}$ denote the vector that has 0's in the coordinates corresponding to rows that weren't picked, and the random field elements in the coordinates that were picked. We use $|A^{(i,(j,k))}| = 2^k$ to refer to the number of coordinates that were not set to 0 (we ignore the fact that for the non-zero-fixed rows, the random field elements picked can also be 0 with small probability, this will have no impact on the analysis).

We can now describe the k-th forked input-claim pair (and its distance bound):

- The new input is a linear combination of $y^{(i,j)}$'s rows, according to the coefficients in $A^{(i+1,(j,k))}$:

$$y^{(i+1,(j,k))} = \sum_{h \in \mathbb{H}} A^{(i+1,(j,k))}[h] \cdot y_h^{(i,j)}$$

- The new claim is derived by taking linear combinations of the prover's claims in this round:

$$\vec{v}^{(i+1,(j,k))} = \sum_{h \in \mathbb{H}} A^{(i+1,(j,k))}[h] \cdot W_h^{(i,j)},$$

  where $J^{(i+1,(j,k))}$ was derived above, by omitting the first coordinate in each location of $J^{(i,j)}$.

- The new claimed distance bound is:

$$\delta^{(i+1,(j,k))} = \delta^{(i,j)} \cdot \frac{2^k}{n_{imp}^{o(1)}}. \tag{13}$$

Completeness follows by linearity of the LDE. The soundness argument is more involved, but it shows that if the original input had the claimed distance bound from its claims, then w.h.p. at least one of the new forked inputs will have the (new) claimed distance from its (new) claims. We refer the reader to [RVW13] for further details.

Deriving the output claim. The number of rounds r is a constant set below. After r rounds, we have a collection of $f^r = n_{imp}^{o(1)}$ input-claim pairs. In the final round, for each $j \in [f]^r$, the prover sends to the verifer (explicitly) a claim $\tilde{y}^{(j)}$ about the value of the final input in the j-th execution path. The communication complexity for sending $\tilde{y}^{(j)}$ shrinks with the number of rounds (since each round reduces the length of the inputs by a $|\mathbb{H}| \approx n^{\delta}$ multiplicative factor). We set r so that the length of these final inputs is $|\mathbb{H}|^{m-r} \approx (\varepsilon \cdot n_{imp}^{1+o(1)})$. To complete the protocol, the verifier runs two final tests for each of the j final input-claim pairs (the j-th pair is indexed by (r,j)):

1. **Alleged-input claim consistency:** the verifier checks that

$$\mathsf{LDE}(\tilde{y}^{(j)})[J^{(r,j)}] = \vec{v}^{(r,j)},$$

   and rejects immediately if this is not the case.

2. **Alleged-input real-input consistency:** the verifier picks a set $B^{(j)} \subset \mathbb{H}^{m-r}$ of random coordinates in $\tilde{y}^{(j)}$ (or rather, pseudo-random coordinates: see Proposition A.4), and checks their consistency with the true implicit input y. This test will be performed by the predicate ψ. We detail its functionality and then describe the circuit structure.

   The set $B^{(j)}$ is of size $b^{(j)} = \mathsf{polylog}(n_{imp})/\delta^{(r,j)}$, i.e. it is chosen so that if $\tilde{y}^{(j)}$ is $\delta^{(r,j)}$-far from the correct vector $y^{(j)}$, then w.h.p. the two will differ on at least one coordinate in $B^{(j)}$. We represent $B^{(j)}$ as a $|\mathbb{H}|^{m-r}$-dimensional vector, with $b^{(j)}$ coordinates that are fixed to 1, and the other coordinates are fixed to 0.

   Now the verifier computes the value of the true reduced input $y^{(j)}$ at each of these coordinates, and compares them with the value in $\tilde{y}^{(j)}$. Unrolling the verifier's choices during the j-th execution path, for each coordinate $\rho \in B^{(j)}$ the verifier checks that:

$$y^{(r,j)}[\rho] = \sum_{\tau \in \mathbb{H}^r} \left(\prod_{i \in [r]} A^{(i,(j_1,\ldots,j_i))}[\tau_i] \right) \cdot y[(\tau,\rho)] = \tilde{y}^{(j)}[\rho]. \tag{14}$$

   Recalling that many of the elements in the $A^{(i,j)}$ and $B^{(j)}$ vectors were fixed to 0, the query complexity for checking input-consistency on the j-th input is:

$$\begin{aligned} q_{exec} &= \left(\prod_{i \in [r]} 2^{j_i} \right) \cdot b^{(j)} \\ &= \left(\prod_{i \in [r]} 2^{j_i} \right) \cdot \frac{\mathsf{polylog}(n_{imp})}{\delta^{(r,j)}} \\ &= \left(\prod_{i \in [r]} 2^{j_i} \right) \cdot \frac{\mathsf{polylog}(n_{imp})}{\delta^{(0,(\cdot))}} \cdot \left(\prod_{i \in [r]} \frac{n_{imp}^{o(1)}}{2^{j_i}} \right) \\ &\leq \frac{n_{imp}^{o(1)}}{\delta^{(0,(\cdot))}} \\ &\approx \frac{1}{\varepsilon} \cdot n_{imp}^{o(1)}, \end{aligned}$$

   where we remark that the per-execution-path query complexity q_{exec} is independent of the execution index j.

Remark A.3. *In this overview, we show the query complexity is $(1/\varepsilon) \cdot n_{imp}^{o(1)}$. RVW actually show $(1/\varepsilon)^{(1+o(1))}$ by an appropriate choice of parameters. In our use of the theorem ε will be $\mathsf{poly}(1/n_{imp})$, so the distinction is moot.*

Before describing the circuits for performing the alleged-input real-input consistency test, we wrap up the analysis of the protocol. In terms of complexity, there are $f^r = n_{imp}^{o(1)}$ execution paths, so the total query complexity, communication complexity, and verifier runtime are as claimed. Completeness follows by construction. Soundness follows because w.h.p. at least one of the final inputs $y^{(r,j)}$ is far from its claim. Either the prover sends $\tilde{y}^{(j)}$ that doesn't satisfy the claim, in

which case the Alleged-input claim consistency test fails, or the prover sends $\tilde{y}^{(j)}$ that satisfies the claim, but then $\tilde{y}^{(j)}$ must be far from the real $y^{(j)}$, and the alleged-input real-input test fails.

As already hinted, the coordinates in this test can be taken to be *pseudo-random*. This is captured by the following proposition.

Proposition A.4 (Derandomization of the set $B^{(j)}$)**.** *Take $n' = |\mathbb{H}|^{m-r}$, and assume that $|y^{(j)}| = |\tilde{y}^{(j)}| = n'$ and that they are at distance $\delta^{(r,j)}$ from each other. Then, there exists a $\mathsf{polylog}(n)$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit of size $\mathsf{polylog}(n)$, such that given a uniformly random string $\xi \in_R \{0,1\}^{(2\log^3(n'))}$ as advice, outputs a set $B^{(j)} \subset \{0,1\}^{\log(n')}$ of size $b^{(j)} = \mathsf{polylog}(n')/\delta^{(r,j)}$ of coordinates in $[n']$, such that with probability at least $1-\lambda$ over the choice of ξ, $y^{(j)}$ and $\tilde{y}^{(j)}$ differ on at least one coordinate in $B^{(j)}$, where $\lambda = \lambda(n) : \mathbb{N} \to \mathbb{N}$ is smaller than any polynomial.*

Proof. The proof consists of two steps: First, using ξ for generating a set of pairwise independent coordinates, and second, showing that pairwise independence is enough for catching a difference between $y^{(j)}$ and $\tilde{y}^{(j)}$ with high probability — namely, that comparing $y^{(j)}$ and $\tilde{y}^{(j)}$ only on the coordinates included in this set is enough for catching w.h.p. an inconsistency between the strings.

Generating $B^{(j)}$. The set $B^{(j)}$ is composed of subsets, such that each *subset* satisfies pairwise independence. These subsets will be used to perform parallel repetitions of the comparison between $y^{(j)}$ and $\tilde{y}^{(j)}$. This will allow us to achieve the required soundness error while only checking $\mathsf{polylog}(n')/\delta^{(r,j)}$ coordinates, and by that save in random bits (which are the length of the advice string), and in the size of the circuit that generates these coordinates, as described below.

We take $C_{B^{(j)}}$ to denote the circuit that generates $B^{(j)}$. Recall that $|B^{(j)}| = b^{(j)}$. Following Definition 3.21, this circuit maps indices in $[b^{(j)}]$ to coordinates in $[n']$. Namely, when given an index $i \in \{0,1\}^{\log b^{(j)}}$, it returns the i^{th} element of $B^{(j)}$, which is a coordinate in $\{0,1\}^{\log(n')}$ (identified with $[n']$). However, we stress that it does not perform the comparisons: The next component we describe — the circuit that computes ψ — is the one that compares $y^{(j)}$ and $\tilde{y}^{(j)}$.

Let $t < n'$ be a parameter to be set later, indicating the size of each pairwise independent subset. We describe the structure of $C_{B^{(j)}}$, then prove that it satisfies the stated size and uniformity properties.

- *Structure.* We use a family of pairwise independent functions that map strings from $\{0,1\}^{\log t}$ to strings in $\{0,1\}^{\log(n')}$. A known construction of such a family uses random linear maps $\{h_{A,b} : \{0,1\}^{\log(n')} \to \{0,1\}^{\log(n')}\}$,[27] such that $h_{A,b}(x) = Ax + b$, where A is a $(\log(n') \times \log(n'))$ Boolean matrix and b is a binary vector of length $\log(n')$. We take $C_{A,b}$ to denote this circuit, as it uses (A,b) as advice string.

  The circuit $C_{B^{(j)}}$ is composed of $\log(n')$ copies of $C_{A,b}$, where it uses a different key (A,b) as the advice string of each copy (these keys will be randomly chosen by the verifier). The copies represent parallel repetitions, thus it is important that we use independent keys. The construction promises that a random (A,b) generate pairwise independent strings $\{h_{A,b}(x)\}_{x \in \{0,1\}^{\log t}}$.

- *Size and uniformity.* First, we find $C_{A,b}$'s size and depth. Given (random) A and b, computing a map from the pairwise independent family requires multiplication of a matrix and a vector, and addition of two vectors. Since these vectors are binary, these two operations can be

[27] Since we take t such that $n' > t$, we pad the input $x \in \{0,1\}^{\log t}$ with zeros, i.e., we apply $h_{A,b}(x \circ 0^{\log(n') - \log t})$.

done by a depth-3 Boolean circuit with parity gates, of size $O(\log^2(n'))$. Thus, $C_{A,b}$ forms an $\mathsf{AC}^0[\oplus]$ circuit.

The circuit is $O(\log\log(n'))$-highly uniform, following the same arguments used in Claim 3.22: the most "expensive" operation (with respect to circuit depth) that the adjacency predicate performs is comparing two binary strings of length $O(\log\log(n'))$, which is the length of each label.

Notice that a random function from this family can be selected using $\log^2(n') + \log(n') \leq 2\log^2(n')$ random bits, referred to as the key (these bits instantiate A and b). This means that $C_{A,b}$ is also $(2\log^2(n'))$-succinct.

Now, recall that $C_{B^{(j)}}$ should map indices $i \in \{0,1\}^{\log b^{(j)}}$ to $B^{(j)}$'s i^{th} index. Thus, it first uses the first $\log\log(n')$ bits of its input to decide which of the copies to use, and then it feeds the selected copy with the rest of the input. It is clear that this operation can be done by a $\log\log(n')$-depth arithmetic circuit over $\mathbb{GF}[2]$.

We define $C_{B^{(j)}}$'s advice string to be the concatenation of all $\log(n')$ advice strings, of length $(2\log^2(n'))$ each, needed for computing each of the copies of $C_{A,b}$. Namely, the advice string of $C_{B^{(j)}}$ is $(2\log^3(n'))$ random bits.

Overall, we conclude that $C_{B^{(j)}}$ is a $(2\log^3(n'))$-succinct $O(\log\log(n'))$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit of size $O(\log^3(n'))$. Recalling that $n' \leq n_{imp}$ and that $\mathsf{polylog}(n_{imp}) \leq \mathsf{polylog}(n)$, we get the stated parameters.

Using $B^{(j)}$ for alleged-input real-input consistency test. Given the $b^{(j)}$ coordinates in $[n']$, we need to prove that the verifier catches an inconsistency between $y^{(j)}$ and $\tilde{y}^{(j)}$ with high probability. Recall that the distance between the strings is at least $\delta^{(r,j)}$, and that $B^{(j)}$ is composed of $\log(n')$ subsets of pairwise independent coordinates, each of size t.

Now, we take X to be a random variable indicating the number of coordinates that are different between the strings, when checking t pairwise independent coordinates. Notice that $\mathrm{E}[X] = t \cdot \delta^{(r,j)}$ and that $\mathrm{Var}[X] \leq t \cdot \delta^{(r,j)}$. Observe that the probability that the strings are equal when checking a single subset is

$$\Pr[X = 0] \leq \Pr\left[\mathrm{E}[X] - X \geq t \cdot \delta^{(r,j)}\right] \leq \Pr\left[|X - \mathrm{E}[X]| \geq t \cdot \delta^{(r,j)}\right],$$

thus, Using Chebyshev's inequality,

$$\Pr[X = 0] \leq \frac{\mathrm{Var}[X]}{(t \cdot \delta^{(r,j)})^2} \leq \frac{1}{\delta^{(r,j)} \cdot t}.$$

Overall, the probability that the strings are equal on $B^{(j)}$ is at most

$$\left(\frac{1}{\delta^{(r,j)} \cdot t}\right)^{\log(n')} = \left(\frac{1}{\log(n')}\right)^{\log(n')},$$

by taking $t = \log(n')/\delta^{(r,j)}$. This error function is smaller than any polynomial in n, as n and n' are polynomially related (since $n' \geq |\mathbb{H}| = n^{\delta}$). To conclude, we define $b^{(j)} = t \cdot \log(n') = \log^2(n')/\delta^{(r,j)}$. □

Highly uniform representation of Q. We are now (finally) ready to describe the circuits C_Q, computing the list representing the query set, and C_ψ, which checks the predicate ψ.

The query set Q is represented by a list, where each element in the list corresponds to a query to a location in the input. Each query location is represented as a pair (τ, ρ) for $\tau \in \mathbb{H}^r$, $\rho \in \mathbb{H}^{m-r}$ (recall that $\rho \in B^{(j)} \subset \mathbb{H}^{m-r}$). The list is indexed by tuples in $[f]^r \times \{0,1\}^{\log(q_{exec})}$, parsed as follows:

- The first part is an execution number $j \in [f]^r$.
- For fixed prefix j, the next part of the suffix is $\alpha \in \{0,1\}^{\sum_{i=1}^{r} j_i}$, which represents a choice of a non-zero location in each set $A^{(i,(j_1,\ldots,j_i))}$ of coordinates chosen in the i-th round of this execution path.
- The last part is $\beta \in \{0,1\}^{\log(b^{(j)})}$, which represents the choice of coordinates of the Alleged-input that will be tested.

Thus, the total length of the list is as claimed. The mapping C_Q of an index (j, α, β) in the list to a location (j, τ, ρ) in the implicit input is done as in Equation (14) above. (j, α, β) should be mapped to the α-th query of the β-th coordinate of $\tilde{y}^{(j)}$ that is checked. The main issues are:

- mapping α to the α-th query location. Towards this, for each execution path j and each round i, the verifier can prepare a small lookup table $T^{(i,j)} : \{0,1\}^{j_i} \to \mathbb{H}$, which maps an index γ to the γ-th row of that execution round's input that was chosen to have a non-zero coefficient. This lookup table is of size at most $|\mathbb{H}|$, and can be computed by an $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit of size $n^{\delta+o(1)}$.

  The first part of the mapping C_Q simply concatenates these $r = O(1)$ circuits. We divide α into r chunks, where the i-th chunk α_i is of length j_i, and output:

$$C^j_{Q,1}(\alpha_1, \ldots, \alpha_r) = \left(T^{(1,j)}(\alpha_1), \ldots, T^{(r,j)}(\alpha_r)\right).$$

- mapping β to the β-th location that is tested in $\tilde{y}^{(j)}$ is simpler: we pick a list of $b^{(j)}$ input locations using Proposition A.4, and output the lookup table that maps β to the β-th location in the list. We obtain a $\mathsf{polylog}(n)$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuit $C^j_{Q,2}$ of size $\mathsf{polylog}(n)$ (namely, we take $C^j_{Q,2}$ to be $C_{B^{(j)}}$ from the proposition), that maps β to the β-th location in the list.

On input (j, α, β), the circuit C_Q outputs $(j, C^j_{Q,1}(\alpha), C^j_{Q,2}(\beta))$. This is performed by using j to do a "lookup" among the $n^{o(1)}_{imp}$ j-specific circuits. Since all of the subcircuits are (at most) $n^{\delta+o(1)}$-succinct $O(\log n)$-highly uniform $\mathsf{AC}^0[\oplus]$ circuits of size $n^{\delta+o(1)}$, we conclude that C_Q satisfies these uniformity and size requirements as well. We stress that the depth of C_Q is independent of δ, i.e., it is some global constant.

Highly uniform representation of ψ. The circuit for ψ proceeds along similar lines. Its inputs are indexed by the list indices (j, α, β). For each execution path j, we need to check the alleged-input real-input claims. The verifier can prepare a lookup table, mapping each j and β to the claimed value $\tilde{y}^{(j)}[C^{(j)}_{Q,2}(\beta)]$, followed by the same arguments from above.

We now need to map the α-th input wire to the appropriate coefficient, to check that Equation (14) holds. This is again done by parsing α in blocks α_i of length i, and having a lookup table $U^{(i,j)}$ for mapping α_i to the appropriate non-zero coefficient in $A^{(i,j)}$.

Finally, the circuit for ψ checks that Equation (14) holds for each j and β, i.e. that:

$$\tilde{y}^{(j)}[C^{(j)}_{Q,2}(\beta)] = \sum_{\alpha_1 \in \{0,1\}^{j_1}, \ldots, \alpha_r \in \{0,1\}^{j^r}} \left(\left(\prod_{i \in [r]} U^{(i,j)}(\alpha_i) \right) \cdot (\psi\text{'s } (j,\alpha,\beta)\text{-th input}) \right) \tag{15}$$

As claimed, these checks can be performed by an arithmetic $n^{\delta+o(1)}$-succinct $\#\mathsf{AC}^0_{\mathbb{F},f_{in}}$ circuit, where the fan-in for multiplication (as above) is logarithmic (this requires using the approximation method of Theorem 3.18 to compute a large conjunction on all the individual tests that we run). Its depth is a global constant, independent of δ. This circuit is also $O(\log n)$-highly uniform, following the same arguments from above.

A.4 Proof of Claim 3.22

We construct a circuit that gets a field element $x \in \mathbb{F}$ and outputs its binary representation $[x]_2$. To facilitate the labeling, we allow gates in all layers to use the constants from the input layer, although the circuit is *layered* and the constants are moved from layer to layer and labeled appropriately.

The circuit has 7 layers, including the input and the output. The label of each gate is of the form (t, ℓ_1, ℓ_2), where $t \in \{0,1\}^3$ denotes the layer (notice that there is no layer 0), and $(\ell_1, \ell_2) \in \{0,1\}^{\log(|\mathbb{F}|)} \times \{0,1\}^{\log\log(|\mathbb{F}|)}$ denote the internal label. It is easier to think of labels and indices as integers, thus we often use the notation $[k]_2$ for the $\log(|\mathbb{F}|)$-bit binary expansion of an integer $k \in [|\mathbb{F}|]$. In most cases, ℓ_2 will be taken to be empty, and in this case, we write $\vec{0}$ instead of $0^{\log\log(|\mathbb{F}|)}$. It will also be convenient to use $\vec{1}$ instead of $1^{\log\log(|\mathbb{F}|)}$.

- **Layer 1:** The first (input) layer is composed of a list of all of the constants, that is, all field elements, in an increasing order ($\mathbb{F}$ is an ordered field). For $0 \le i \le |\mathbb{F}| - 1$, the label of f_i is $(001, [i]_2, \vec{0})$. The last gate of this layer is the circuit's input, x, and its label is $(001, [0]_2, \vec{1})$.
- **Layer 2:** The second layer has $|\mathbb{F}|$ addition gates, labeled $(010, [i]_2, \vec{0})$ for $0 \le i \le |\mathbb{F}| - 1$. The i^{th} gate computes $(x + f_i)$, which is also $(x - f_i)$, and we will prefer the latter notation.
- **Layer 3:** The third layer has the following $|\mathbb{F}|$ multiplication gates, labeled $(011, [i]_2, \vec{0})$ for $0 \le i \le |\mathbb{F}| - 1$. The i^{th} gate outputs:

  $$\prod_{f \in \mathbb{F}, f \neq f_i} (x - f).$$

  Moreover, it has an additional multiplication gate, labeled $(011, [0]_2, \vec{1})$, that computes

  $$\prod_{f \in \mathbb{F}, f \neq f_0} f.$$

  Notice that for every $0 \le i \le |\mathbb{F}| - 1$, this result is the same as $\prod_{f \in \mathbb{F}, f \neq f_i} (f_i - f)^{-1}$, namely, the product of all invertible (non-zero) elements is also the product of the inverses of all invertible elements. Thus, the last gate avoids computing inverses, but rather perform a much simpler computation that can be done within the uniformity demands that we want.

- **Layer 4:** The forth layer multiplies the first $|\mathbb{F}|$ gates with the last one. For $0 \leq i \leq |\mathbb{F}| - 1$, the i^{th} gate of this layer is labeled $(100, [i]_2, \vec{0})$ and it computes:

$$\prod_{f \in \mathbb{F}, f \neq f_i} \frac{x - f}{f_i - f}.$$

- **Layer 5:** The fifth layer adds the constant $f_1 = 1$ to each of the results. This means that the i^{th} gate outputs 1 if $x = f_i$, and 0 otherwise. Again, we label them by $(101, [i]_2, \vec{0})_{0 \leq i \leq |\mathbb{F}|-1}$.

- **Layer 6:** The sixth layer uses $(|\mathbb{F}| \cdot \log(|\mathbb{F}|))$ multiplication gates for multiplying the i^{th} result of the fifth layer described above (that will have fan-out of $\log(|\mathbb{F}|)$) with $[f_i]_2$, that is, the i^{th} sequence of the binary representations of all field elements (computed by the incidence function, detailed next). This means that for $x \neq f_i$, we get a $(\log(|\mathbb{F}|))$-long sequence of 0's, and for $x = f_i$, we get $[f_i]_2$. The labeling of this layer goes as follows: The j^{th} gate of the i^{th} sequence, where $0 \leq i \leq |\mathbb{F}| - 1$ and $0 \leq j \leq \log(|\mathbb{F}|) - 1$, is $(110, [i]_2, [j]_2)$.

- **Layer 7:** The seventh (output) layer is the bitwise sum of these $\log(|\mathbb{F}|)$ results, i.e., $\log(|\mathbb{F}|)$ addition gates of fan-in $|\mathbb{F}|$, labeled $(111, [i]_2, \vec{0})_{0 \leq i \leq \log(|\mathbb{F}|)-1}$. Since a sum of a bit and 0's returns the bit, the circuit outputs $[x]_2$.

It is evident that this circuit is an $\#\mathsf{AC}^0_{\mathbb{F}, f_{in}}$ circuit with multiplication gates of fan-in $|\mathbb{F}|$. Next, we prove that it is $O(\log n)$-highly uniform. With respect to the labeling defined above, we construct the adjacency predicate for the circuit's addition gates, and incidence function for the circuit's multiplication gates. Recall that each of them should be computed by an arithmetic circuit of size at most $n^{\delta + o(1)}$ and logarithmic degree. In fact, we will show that they can be computed by circuits of polylogarithmic size.

Denoting the labels' length by $\mu = 3 + \log(|\mathbb{F}|) + \log\log(|\mathbb{F}|)$, we take

$$\phi_{adj} : \{0,1\}^{\mu} \times \{0,1\}^{\mu} \to \{0,1\}$$

to be the circuit's adjacency predicate, and

$$\phi_{incd} : \{0,1\}^{\mu} \times \{0,1\}^{\log(|\mathbb{F}|)} \to \{0,1\}^{\mu}$$

to be the circuit's incidence function. For transparency, we present their functionalities layer-by-layer. Layers 2, 5 and 7 are of addition gates, therefore will be handled by ϕ_{adj}, whereas layers 3,4 and 6 are of multiplication gates, therefore will be handled by ϕ_{incd}. We always assume that the circuits should return 0 or 0^{μ} for all possible labels that are not explicitly mentioned. Since the first layer has no incoming wires, we start with the second layer.

- **Layer 2:** For $0 \leq i \leq |\mathbb{F}| - 1$, it should hold that

$$\phi_{adj}\left((010, [i]_2, \vec{0}), (001, [0]_2, \vec{1})\right) = 1,$$

and

$$\phi_{adj}\left((010, [i]_2, \vec{0}), (001, [i]_2, \vec{0})\right) = 1.$$

For the second condition, the circuit should compare the middle part of each label (the one corresponding to ℓ_1). A comparison of two binary strings y_1, y_2 of length $\log(|\mathbb{F}|)$ by an arithmetic circuit over $\mathbb{GF}[2]$ can be done as follows:

$$\prod_{i\in[|y_1|]} \left((1-y_1[i])+y_2[i]\right)\cdot\left(y_1[i]+(1-y_2[i])\right),$$

while recalling that subtraction is the same as addition. The size and the degree of this circuit is $O(\log(|\mathbb{F}|)) = O(\log n)$. For the first condition and for finding the type t in both conditions, the circuit of ϕ_{adj} perform a straightforward comparison between the labels and constants, using the same argument.

- **Layer 3:** For $0 \le i \le |\mathbb{F}|-1$, the incidence function should output

$$\phi_{incd}\left((011,[i]_2,\vec{0}),[j]_2\right) = \begin{cases} (010,[j]_2,\vec{0}) & j<i \\ (010,[j+1]_2,\vec{0}) & j\ge i \end{cases},$$

and

$$\phi_{incd}\left((011,[0]_2,\vec{1}),[j]_2\right) = (000,[j]_2,\vec{0}).$$

A circuit that computes the second condition is again a simple comparison to constant bits, as described in the previous item. The first condition, however, requires more care.

For two bits b, b', we define $(b<b') \stackrel{\text{def}}{=} ((1-b)\cdot b')$ and $(b=b') \stackrel{\text{def}}{=} ((1-b)+b')\cdot(b+(1-b'))$. We denote $[j]_2 = (j_{\log(|\mathbb{F}|)-1},...,j_2,j_1,j_0)$, and $[i]_2$ analogously. First, the circuit computes:

$$\begin{aligned} res \stackrel{\text{def}}{=} &(j_0<i_0)+((j_0=i_0)\cdot(j_1<i_1))+((j_0=i_0)\cdot(j_1=i_1)\cdot(j_2<i_2))+...+ \\ &\left((j_0=i_0)\cdot...\cdot(j_{\log(|\mathbb{F}|)-1}=i_{\log(|\mathbb{F}|)-1})\right). \end{aligned}$$

The circuit's output is as follows. The first $(\log(|\mathbb{F}|)-2)$ output bits are $(j_{\log(|\mathbb{F}|)-1},...,j_2)$. For the last two bits, the circuit compares the result bit res (notice that the circuit must duplicate res for this comparison) and 0:

  - If $res = 0$, the last two output bits are (j_1, j_0);
  - If $res = 1$ and $j_0 = 0$, the last two output bits are $(j_1, 1)$;
  - If $res = 1$ and $j_0 = 1$, the last two output bits are $(1, 0)$.

The comparison is done as described above, and the if condition is done by a product of the comparison and the appropriate output.

- **Layer 4:** For $0 \le i \le |\mathbb{F}|-1$, the incidence function should output

$$\phi_{incd}\left((100,[i]_2,\vec{0}),[1]_2\right) = (011,[i]_2,\vec{0}),$$

and

$$\phi_{incd}\left((100,[i]_2,\vec{0}),[2]_2\right) = (011,[0]_2,\vec{1}).$$

This is done in the same manner as in the previous item.

- **Layer 5:** The adjacency predicate should satisfy the following two conditions:

$$\phi_{adj}\left((101,[i]_2,\vec{0}),(001,[1]_2,\vec{0})\right)=1,$$

for adding the constant 1, and

$$\phi_{adj}\left((101,[i]_2,\vec{0}),(100,[i]_2,\vec{0})\right)=1,$$

to connect the result of the previous layer. The circuit that computes them works exactly like the one constructed for the second layer.

- **Layer 6:** For $0 \leq i \leq |\mathbb{F}|-1$ and $0 \leq j \leq \log(|\mathbb{F}|)-1$, we denote

$$\alpha_{i,j} \stackrel{\text{def}}{=} \text{the } j^{\text{th}} \text{ bit of } [f_i]_2.$$

The incidence function computes:

$$\phi_{incd}\left((110,[i]_2,[j]_2),[1]_2\right)=(001,[\alpha_{i,j}]_2,\vec{0}),$$

and

$$\phi_{incd}\left((110,[i]_2,[j]_2),[2]_2\right)=(101,[i]_2,\vec{0}).$$

Whereas the second condition can be implement in the same manner as before, we need to specify a circuit for the first one, in order to find $\alpha_{i,j}$.

First, notice that following the definition of a binary representation of a field elements, $[f_i]_2 = [i]_2$. Taking $y=[i]_2$, this means that ϕ_{incd} gets two binary strings $(y,[j]_2)$ as input, such that the second represents an index, and outputs $y[j]$. An arithmetic circuit over $\mathbb{GF}[2]$ with the same functionality was already constructed in Section 4.4.1, under the name C_{select}. We use this circuit with $|z_1|=1$ and $k=|y|=\log(|\mathbb{F}|)$ (as done in Lemma 4.4, when proving the uniformity of ψ'). The circuit C_{select} is of size $\tilde{O}(\log(|\mathbb{F}|))$ and degree $f_{in}=\log\log(|\mathbb{F}|)$.

- **Layer 7:** For $0 \leq i \leq |\mathbb{F}|-1$ and $0 \leq j \leq \log(|\mathbb{F}|)-1$, the adjacency predicate computes:

$$\phi_{adj}\left((111,[i]_2,\vec{0}),(110,[i]_2,[j]_2)\right)=1.$$

As we already proved, comparing two strings of length $\log(|\mathbb{F}|)$ can be done using an arithmetic circuit over $\mathbb{GF}[2]$ of logarithmic size.

To conclude, we take the output of each of the full circuits to be the summation of all of the outputs of corresponding subcircuits. We get that each of the final circuits for ϕ_{adj} and ϕ_{incd} are of size $\mathsf{polylog}(n)$ and degree $O(\log n)$, and the claim follows.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il