Electronic Colloquium on Computational Complexity, Report No. 111 (2026)

Doubly-Efficient Interactive Arguments for Bounded-Space from One-Way Functions

Guy N. Rothblum*
Apple

Abstract

We show that one-way functions suffice for constructing very efficient argument systems for proving the correctness of bounded-space computations. Taking κ to be a cryptographic security parameter and n to be the input length, our argument system applies to general computations running in time T and space S. The protocol has $\mathsf{polylog}(T)$ rounds, the communication complexity is $S \cdot \mathsf{poly}(\log(T), \kappa)$ and the verifier runtime is $(S+n) \cdot \mathsf{poly}(\log(T), \kappa)$. In particular, the complexity of the verifier is only poly-logarithmic in the computation length. The honest prover runs in time $\mathsf{poly}(T, \kappa)$, so the protocol is doubly-efficient. If the one-way function is secure against $\lambda(\kappa)$-size adversaries, then the argument system is sound against cheating provers of size $\lambda(\kappa)/\mathsf{poly}(T)$.

Prior to this work, doubly-efficient argument systems with poly-logarithmic dependence on T required assuming (at least) the existence of collision-resistant hash functions [Kilian, STOC 1992]. For unconditionally sound doubly-efficient protocols, the RRR protocol for bounded-space computations [Reingold, Rothblum and Rothblum, STOC 2016] has communication and verification complexities that grow with T^{σ} for an arbitrarily small constant $\sigma > 0$.

*Email: rothblum@alum.mit.edu.

ISSN 1433-8092

Contents

1 Introduction **1**
- 1.1 Our Results . . . 2
- 1.2 Further Related Work . . . 3

2 Technical Overview **4**
- 2.1 Recursive Honing Protocol . . . 5
- 2.2 Improvements to Unconditional DEIPs . . . 11

3 Preliminaries and Definitions **13**
- 3.1 Cryptographic Security, UOWHFs and Hash Trees . . . 14
- 3.2 Succinct Descriptions . . . 16
- 3.3 Reversible Computation, Tableau and Multi-Tableau . . . 17
- 3.4 Polynomials, Low Degree Extensions and Checksums . . . 19
- 3.5 Holographic Interactive Proofs and Arguments . . . 22
- 3.6 Interactive Proofs of Proximity . . . 24
- 3.7 Honing Protocols . . . 30

4 Main Construction **32**
- 4.1 Base of The Recursion . . . 32
- 4.2 The Recursive Step . . . 33
- 4.3 The Resulting Honing Protocol . . . 50
- 4.4 Arguments for Bounded Space . . . 54

5 Improved Unconditional DEIP **56**
- 5.1 Base of The Recursion . . . 56
- 5.2 The Recursive Step . . . 57
- 5.3 The Resulting Honing Protocol . . . 61
- 5.4 The Resulting DEIP . . . 64

A The AR Argument System **67**

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 this work, we focus on proof systems that take the form of an interactive challenge-response protocol between the prover and the verifier. Such a protocol can be unconditionally sound against any (unbounded) cheating prover, i.e. an interactive *proof* system [GMR89]. Computationally sound *argument* systems [BCC88] relax soundness to hold only against computationally bounded cheating provers and under cryptographic assumptions.

In this work, we study the resources and the computational assumptions needed to prove and to verify general claims. Our starting point is a simple and appealing cut-and-choose protocol. Suppose the prover claims that the input is in a language that can be decided by a Turing Machine running in time T and space S. In each round of the cut-and-choose protocol, the prover sends the mid-point state of the computation currently under consideration, and the verifier chooses at random whether to check the first or the second half of the computation. In more detail: initially the computation is of length T, the prover and the verifier agree on the initial state u_0, and the prover claims that the final state is $\widetilde{u_T}$. The prover sends the alleged mid-point of the computation $\widetilde{u_{T/2}}$, and the verifier decides at random whether to check the first or second half. The protocol then recurses whichever half of the computation the verifier chose. After $\log(T)$ rounds, at the base of the recursion, the claim is about a computation of length 1, and the verifier can verify this claim on its own. The ideas underlying this protocol are important to foundational results in complexity theory, most notably the proof that the language TQBF is PSPACE-complete and the study of alternating computation [CKS81]. The proof-system is incredibly efficient: the honest prover runs in time $O(T)$ and essentially just needs to run the computation. The communication complexity is only $O(S \cdot \log(T))$ and there are only $O(\log(T))$ rounds. The verifier's runtime is $O(S \cdot \log(T) + n)$ (in the base of the recursion, the verifier needs to read the input to perform a single step of the computation). The main issue is that the soundness error of the simple cut-and-choose protocol is huge: in each round, if the current claim is false, a cheating prover can decide at random whether to cheat on the first or the second half of the computation. With probability $1/2$ the verifier will recurse on the "wrong" half (the half where the prover's claim is true), and from this point on the prover can convince the verifier with probability 1. Thus, the verifier's probability of catching a cheating prover is only $(1/T)$: the soundness error is $1 - (1/T)$. We study the following question:

Question 1.1. *Under what assumptions can we construct a proof system with negligible soundness error that matches, up to polynomial factors, the complexities of the cut-and-choose protocol?*

In particular, we focus on *doubly-efficient* proof systems (DEIPs) [GKR15], where the honest prover's running time should be $\mathsf{poly}(T)$ (as it is in cut-and-choose). Reingold, Rothblum and Rothblum [RRR16] constructed unconditionally sound DEIPs: for an arbitrarily small constant power $\sigma > 0$, their proof system has $T^{\sigma} \cdot \mathsf{poly}(S)$ communication, $(T^{\sigma} \cdot \mathsf{poly}(S) + \widetilde{O}(n))$ verifier runtime and constant round complexity (the round complexity is quasi-exponential in $(1/\sigma)$). Question 1.1, however, asks for doubly-efficient proofs whose communication and verification time are *poly-logarithmic* in T. No such result is known for unconditionally sound protocols.

Thus, we turn our attention to doubly-efficient *argument* systems, which are sound against computationally bounded cheating provers and under cryptographic assumptions. The weakest cryptographic assumption under which previous work gives a positive answer to Question 1.1 is the existence of collision-resistant hash functions (CRH). Under this assumption, Kilian's argument

system [Kil92] matches the cut-and-choose protocol, and even goes beyond it. Taking κ to be a cryptographic security parameter, the communication is $\mathsf{poly}(\log(T), \kappa)$ and the verifier's runtime is $(\mathsf{poly}(\log(T), \kappa) + \widetilde{O}(n))$. Indeed, there is no dependence on the space and the protocol also applies to non-deterministic computations. For an argument system, we expect the communication and verifier runtime to be polynomial in the security parameter, so we view this as "matching" (or surpassing) cut-and-choose. Thus, CRHs suffice for answering Question 1.1 in the affirmative, but what about weaker assumptions? In particular, could one-way functions (OWFs), often referred to as a "minimal" assumption for cryptography, suffice?

1.1 Our Results

Our main result shows that one-way functions are sufficient for constructing an argument system with negligible soundness error that matches cut-and-choose up to polynomial factors:

Theorem 1.2. *Suppose that one-way functions exist. Let $\mathcal{L}$ be a language that can be decided in time $T = T(n) \geq n$ and space $S = S(n)$. Let $\kappa = \kappa(n) \leq T$ be a cryptographic security parameter.*

$\mathcal{L}$ has an $\mathsf{polylog}(T)$-round public-coin interactive argument. The communication complexity is $S \cdot \mathsf{poly}(\log(T), \kappa)$. The honest prover runs in time $\mathsf{poly}(T, \kappa)$ and the verifier runs in time $((S+n) \cdot \mathsf{poly}(\log(T), \kappa))$. The protocol has perfect completeness. If the one-way function is secure against $\lambda(\kappa)$-size adversaries, then the protocol is sound against $(\lambda(\kappa)/\mathsf{poly}(T))$-size cheating provers.

See Theorem 4.22 for a fuller statement. Comparing our result to Kilian's protocol, we only assume the existence of OWFs (rather than CRHs). On the other hand, our round complexity is poly-logarithmic (rather than constant), our communication and verifier time have a linear dependence on the space used by the computation, and our protocol doesn't apply to non-deterministic computations. For the class of deterministic computations using space $S = \mathsf{poly}(\kappa)$ our communication and verifier runtime are polynomially related to those achieved by Kilian's protocol.

Comparing our result to the RRR protocol, the communication and verifier runtime in our protocol have a polylogarithmic dependence on T, rather than T^σ for an arbitrarily small constant power σ in RRR. Thus, our protocol answers Question 1.1 in the affirmative, whereas RRR (and all known protocols with unconditional soundness) do not. On the other hand, our round complexity is $\mathsf{polylog}(T)$ (rather than constant) and our protocol is only computationally sound (assuming OWFs). While the results of RRR were stated for an arbitrarily small constant power $\sigma > 0$, their work, and the recent improvement in Berger *et al.* [BGHK25], give results in the sub-constant regime. In this regime, the rounds, communication and verifier runtime grow with $\mathsf{exp}(\widetilde{O}(\log^c(T)))$ for a constant $c \in (0, 1)$. In another recent work, Amit and Rothblum [AR24] constructed constant-round argument systems for bounded-space computations assuming the existence of one-way functions. Similarly to the RRR protocol, the verifier runtime and communication complexity are T^σ for an arbitrarily small constant power σ, whereas we aim for (and achieve) polylogarithmic dependence on T. The advantage of the [AR24] protocol (compared to [RRR16]) was an exponential improvement in the dependence of the round complexity on $1/\sigma$.

Setting the security parameter. Similarly to Kilian's protocol (and as expected for an argument system), the communication and verifier runtime in our protocol are polynomial in the security parameter. We prefer to treat the security parameter, the input length and the computation time as separate parameters. Indeed, one can view the security parameter as a "fixed" parameter, whereas the input and the computation length can grow (this is the view in many more applied works).

Nonetheless, taking n to be the input length, it is also natural to think about adversaries that run in $\mathsf{poly}(n)$ time and set the security parameter appropriately. With this approach in mind, if we assume that the one-way functions have sub-exponential security, then we can set $\kappa = \mathsf{polylog}(n)$ and Theorem 1.2 gives arguments with $\mathsf{polylog}(n)$ communication for the class of $\mathsf{poly}(n)$-time and $\mathsf{polylog}(n)$-space computations. This is the first polylogarithmic proof system for this class that assumes only subexponential OWFs (rather than subexponential CRHs in [Kil92]). Alternatively, if we assume only the existence of polynomially hard OWFs, we can set $\kappa = n^{\varepsilon}$ for arbitrarily small constant $\varepsilon > 0$. For $\mathsf{poly}(n)$-time computations, the communication can be $\widetilde{O}(S \cdot n^{O(\varepsilon)})$. The complexities do not grow with $1/\varepsilon$, whereas in prior works [RRR16, AR24, BGHK25] achieving $\widetilde{O}(S \cdot n^{O(\sigma)})$ communication for a small constant σ, the verifier runtime and communication are (at least) exponential in $(1/\sigma)$ (in [RRR16, BGHK25] the round complexity, while constant, grows exponentially with $(1/\sigma)$. In [AR24] the round complexity only grow polynomially with $(1/\sigma)$).

Improvements to unconditional DEIPs. Along the way, our new techniques also imply an improved construction of information-theoretically sound DEIPs for space-bounded computations: for a computation running in time T and space S, the communication, rounds and verification time grow with $S \cdot \mathsf{exp}(O(\sqrt{\log T}))$. This improves on prior work that achieved $\mathsf{exp}(\widetilde{O}(\log^c(T)))$ dependence for a constants $c > 1/2$ [BGHK25] and matches the state of the art verification complexity, which was recently obtained in an independent work of Chen *et al.* [CHKX26]. See the overview in Section 2.2. The full statement is in Theorem 5.5 and the construction is detailed in Section 5.

Organization. We discuss related work in Section 1.2. Moving beyond the introduction, we highlight several ideas from our construction in a technical overview in Section 2. Preliminaries and definitions are in Section 3, including new definitions of honing protocols and nested IPPs. Our recursive honing protocol construction, and its application to constructing argument systems, are in Section 4. Section 5 details the improved construction of unconditional DEIPs for bounded-space computations.

1.2 Further Related Work

Comparison to known (unconditionally sound) proof systems. We compared our results to [RRR16, BGHK25] above. The celebrated $\mathsf{IP} = \mathsf{PSPACE}$ Theorem [LFKN92, Sha92] gives proof systems for general computations with a $\mathsf{poly}(S, \log(T))$-time verifier (matching cut and choose). The honest prover runtime, however, is super-polynomial in T (also, the round complexity was $\mathsf{poly}(S)$). We focus on doubly-efficient proof systems and require $\mathsf{poly}(T)$-honest prover runtime. The GKR proof system [GKR15] works for computations described by uniform circuits of bounded depth, whereas we are interested in computations described by Turing machines with bounded space. Bounded-space Turing Machines can be translated into bounded-depth circuits, but this incurs an overhead that is exponential in the space complexity, and would not give a $\mathsf{poly}(T)$-time honest prover. Goldreich and Rothblum [GR20] give a constant-round proof system for NC^1 circuits and for lower complexity classes, under suitable notions of uniformity.

Comparison to known (computationally sound) argument systems. We compared our results to [Kil92, AR24] above. Amit and Rothblum [AR23] showed that one-way functions are sufficient for constructing *constant-round* argument systems for uniform bounded-depth circuits,

whereas (similarly to the comparison to [GKR15]) we are interested in bounded-space Turing Machine computations. We note, however, that we use the [AR23] protocol in our construction, and we also build on techniques pioneered in [AR23, AR24].

Under stronger assumptions, arguments exist even beyond NP, and some require only 2 messages or even no interaction at all (starting with Micali's CS proofs [Mic94], see also Kalai, Raz and Rothblum [KRR22] and Choudhuri, Jain and Jin [CJJ21], as well as the expositions by Thaler [Tha22] and Ishai [Ish20a, Ish20b], and the references therein). 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.

2 Technical Overview

For a Turing Machine (TM) $\mathcal{M}$ and an input x (fixed throughout), a *configuration* of the machine specifies the complete state of the machine at an intermediate computation point: it includes the contents of all memory tapes, the TM's internal state and the location of the input reading head. For a machine with space bound S, its configuration can be described using $O(S)$ bits. A *tableau* of $\mathcal{M}$ on x, starting at a configuration u and proceeding for ℓ steps, is the sequence of configurations reached by the machine when its state is set to u and it is run for ℓ steps. Note that we are interested in tableau that start at arbitrary configurations, not just the *starting* configuration u_{start}.[1]

Honing protocols. Honing protocols, which we define and study, are central in our construction. A honing protocol is an interactive protocol between an untrusted prover and a verifier. Given a collection of computations and a claim about their tableaux, the goal is to produce a new claim that "hones in" on (the tableaux of) a smaller subset of the computations. The protocol receives as input a table U containing N initial states, where each state is the beginning of a (long) computation of length ℓ. While all computations are performed using the same Turing Machine and on the same input, they differ in their initial states. The verifier does not access the table U during the protocol and should run in sublinear time in N and in ℓ. The input also specifies succinct predicate ϕ that is claimed to hold for the concatenation of all the tableaux of these computations (the concatenation of the tableaux contains $N \times \ell$ states). The protocol's output is a smaller output subset $B \subseteq [|U|]$ of "surviving" states in U, as well as an output predicate ψ that is claimed to hold for the concatenation of the tableaux of computations in $U|_B$. The protocol's honing parameter η is the fractional size of B within $[U]$, i.e. the fraction of surviving states. Actually, in the body of the paper we use η to denote a bound on the *expected* fraction of surviving rows (the expectation is over the verifier's coins), but we ignore this throughout the overview.

For a certain distance measure, defined below, if the table U is *far* from inducing tableaux that satisfy the predicate, then w.h.p. the tableaux of the computations in the output subset should fail to satisfy the output predicate. The aforementioned distance (measuring "farness") is taken over the table of initial states: being δ-far from satisfying the predicate means that a δ-fraction of the initial states would need to be changed in order for the concatenation of the modified states'

[1] u_{start} is the one where the memory tapes are blank, the internal state is in the initial state, and the reading head is on the first bit of the input.

tableaux to satisfy the predicate (we emphasize that the distance is *not* over the tableaux: we always consider the correct tableaux computed from the given table of initial states). The verifier in a honing protocol "makes progress" because the output claim pertains to a computation of smaller total size ($|B| \cdot \ell$) than the input claim ($|U| \cdot \ell$).

Honing protocols and IPPs. A honing protocol can be viewed as an Interactive Proof (or Argument) of Proximity (an IPP or IAP) over the table of states: a protocol where the verifier only needs to read a subset of the input to decide whether to accept or reject. We find this framing to be helpful, and we use IAPs to construct honing protocols. However, unlike most IPPs, while the *output* claim is only over a sublinear subset of the input, it can be quite computationally complex: it pertains to long computations induced by states in that subset. See also Remark 3.37.

From honing protocols to arguments. While we find honing protocols to be interesting objects in their own right, their study is motivated by applications to constructing argument systems. Suppose we have built a honing protocol with distance δ and honing parameter $\eta \leq 1/2$ for computations of length ℓ. We can use it to build an efficient argument system for computations of length $T \approx \ell$ as follows: the prover sends to the verifier a table U containing $\mathsf{polylog}(T)$ midpoints of the computation: the i-th state in U should be the configuration reached by a computation of length $((i-1) \cdot (T/\mathsf{polylog}(T)))$ starting at u_{start}. The predicate ϕ for the honing protocol checks that: (i) the first state is u_{start}, (ii) the final state in the last tableau is the accepting state, and (iii) the tableaux induced by the states in U describe a single consecutive computation: the first configuration in the $(i+1)$-th tableau proceeds the last configuration in the i-th tableau. If the prover is honest, then the table U should satisfy the predicate ϕ. On the other hand, if the prover is cheating then the computation should reject and there is no possible table that would satisfy ϕ: in particular the table sent by the prover is far (at "infinite" distance) from satisfying ϕ. The prover and verifier now run the honing protocol on U. They end up with an output predicate ψ on a subset B of at most half the states. Intuitively, progress has been made because if the prover was cheating, we now have a false claim about a sequence of tableaux of total length $(T/2)$ (rather than T). The prover and the verifier can now iterate on this new claim. We need to take care to maintain the invariant that in each iteration, the new input is *far* from satisfying the new predicate, and that the complexity of the predicates does not explode: see Section 4.4 for further details.

2.1 Recursive Honing Protocol

We use a recursive construction to obtain honing protocols for increasingly large computation lengths. Taking N to be the input table size and ℓ to be the computation length, we use $T = N \cdot \ell$ to denote the total size of the computation induced by all the rows of the table. We focus on protocols where the communication and verifier run time are poly-logarithmic in T and on the parameter regime where the distance $\delta = O(\mathsf{polylog}(T)/|U|)$ is very small (in much of the body of the paper we work with *absolute distance* $\delta_{\mathsf{ABS}} = \mathsf{polylog}(T)$). A protocol's *honing parameter* η is the fractional size of the output subset of table states (this is analogous to the relative query complexity of an IPP). We aim for honing parameter of order $1/\mathsf{polylog}(T)$.

Base of the recursion, $\ell = \mathsf{polylog}(T)$. At the base of the recursion we use a standard IPP or IAP to construct the "base" honing protocol. We need a protocol with the property that the verifier

does not need to read the input during the protocol: after interacting with the prover, it produces an output predicate over a subset of its input. If the prover was honest then the predicate will be true, otherwise it will be false (in the case of an IAP the latter guarantee is only for bounded adversaries). Many protocols have this form [RVW13, RR20, AR23], as does any protocol in the pre-coordinated model of Goldreich, Rothblum and Skverer [GRS23].

The construction is as follows: for an input U and predicate ϕ for the honing protocol, we define a circuit C that takes the table U as input, computes (in parallel) the ℓ-length tableau for each state in U, and then applies the predicate ϕ. We run an IPP that checks that U (approximately) satisfies the circuit C. Completeness and soundness follow by construction. It was important that the computation lengths be small, because this affects the depth of the circuit C (its depth will be $\widetilde{O}(\ell) + \mathsf{depth}(\phi)$), which shows up in the communication complexity and verifier runtime of known protocols. In particular, we use the IAP of [AR23] to achieve honing parameter $O(1/\mathsf{polylog}(T))$ and $(S \cdot \mathsf{poly}(\kappa, \log(T)))$ communication (the advantage of that construction is in smaller round complexity, see Section 3.6). We remark that we assume here that $|U| \leq \mathsf{poly}(T)$, which will be case in the base of the recursion (more generally, the complexity is poly-logarithmic in $|U|$).

From length ℓ to length $(k \cdot \ell)$: naive attempt. Assume we have a honing protocol for computations of length ℓ. We would like to construct a protocol for computations of length $k \cdot \ell$. We begin with a naive (and unsuccessful) but instructive approach. The idea is to take each of the N input computations of length $(k \cdot \ell)$ and split it into k consecutive computations of length ℓ. For each state U_i in the original input table, the new input table U' contains a block of k states, corresponding to each ℓ-step sub-computation in the original $(k \cdot \ell)$-step computation (i.e. the i-th block contains U_i and the configurations reached by making $\ell, 2\ell, \ldots, ((k-1) \cdot \ell)$ steps from U_i).

We also adapt the predicate. The input to the original predicate ϕ was the concatenation of N tableaux, each of length $(k \cdot \ell)$. The input to the new predicate ϕ' is the new and larger collection of shorter tableaux. The new predicate ϕ' checks two things: first, it enforces "internal" consistency inside each block by verifying that the initial state of each ℓ-step sub-computation is the result of applying a single computation step to the last state in the previous sub-computation (we emphasize that ϕ' gets as input the complete tableau of length-ℓ computations from each state in U', so it can access the last state in each length-ℓ sub-computation directly). Second, ϕ' checks that the original predicate ϕ would accept the concatenation of all length-ℓ tableaux for the states in U'. The additional "internal" consistency checks are lightweight and only add minimal complexity.

We show that if the original input U was δ-far from satisfying ϕ, the new input U' is also δ-far from satisfying ϕ'. Note that this is not completely obvious: e.g. it could be that changing the state U_i to a value v would make ϕ accept (U was at absolute distance 1), but the tableaux induced by U_i and by v are identical except for the initial state, so the "expanded' input U' is also at absolute distance 1 from making ϕ' accept (and the relative distance has degraded, because U' is a larger input). We resolve concern this by considering computations where such "configuration-collisions" shouldn't happen. In particular, we assume the computation is reversible, and show this guarantees that distances are maintained. See Section 3.3 and in particular Fact 3.12 and Remark 3.13.

Thus, we can recursively apply a honing protocol for length-ℓ computations to the new, larger instance (U', ϕ'). The recursive call outputs a subset B' of the states in U' with fractional size η, and a predicate ψ' that detects violation within the length-ℓ tableaux of this smaller subset. Going back to the original input U, we could now output a subset B containing each state in U whose corresponding block in U' contains a state that "survived" in B' (and also output essentially the

same predicate ψ', applied to the length-$(k \cdot \ell)$ tableaux of states in B). The failure point of this approach is that for any block of k states in U' where at least one of the states "survives", that entire block also "survives" in the original set U. This means that a $(k \cdot \eta)$ fraction of the states of U might survive, a k-fold degradation in the honing parameter that makes the entire recursion fail.

Our main technical contribution is a different recursive construction, which maintains the honing parameter, while also making only a single recursive call and adding minimal communication, verifier runtime, and complexity to the predicates being checked.

The tableau UOWHF tree. An important tool in our recursive construction is a hash tree, built from Universal One-Way Hash Functions (UOWHFs) [NY89]. The tree functions as a succinct commitment to a long string (sometimes referred to as a vector commitment). In our case, the string is an encoding of the tableau of a computation induced by a given initial configuration. After the initial configuration is fixed, the receiver / verifier (who need not know the initial configuration or the tableau it induces) sends randomly chosen hash keys. The (honest) sender / prover sends back the correct root of a hash tree built on the encoded tableau. An honest prover, who has committed to the correct root of the hash tree, can then convincingly and succinctly open any location in the encoding of the tableau. A dishonest prover *who sends the correct hash root* for the encoding of the tableau induced by the fixed initial state, cannot later equivocate and open any location to a value different from that of the encoded tableau (at least not without breaking the underlying cryptographic assumption). We emphasize that this guarantee only applies *if the dishonest prover chooses to send the correct hash root*: a dishonest prover can send a false root, and in this case it can later equivocate in its openings. Thus, this is a much weaker guarantee than we could get using full-blown collision-resistant hash functions. However, even this weaker guarantee is helpful: it forces a cheating prover to choose, either it sends a cheating hash root w.r.t. a fixed encoded tableau (and perhaps the verifier can detect this, as below), or it binds itself and its openings have to be consistent with the (encoding of the) correct tableau. Succinct commitments based on UOWHF hash trees were also used in [AR23, AR24] to construct argument systems.

Holographic proofs. Holographic interactive proofs (or arguments) allow a sublinear verifier, who has query access to an input encoded using an appropriate error-correcting code, to verify complex properties of the input *in poly-logarithmic time*. Holographic proof systems were introduced by Babai *et al.* [BFLS91] in the context of PCPs. Gur and Rothblum [GR17] formalized and studied generalizations in the context of interactive proof systems. As examples, in this work we use such protocols to verify that the input is the correct encoding of a certain tableau, or that a given hash root is the correct root of the tree built on the tableau encoded in the input. We use an argument system due to Amit and Rothblum [AR23]: it achieves poly-logarithmic round complexity, with communication and verifier runtime that are polynomial in the security parameter, linear in the circuit depth, and only poly-logarithmic in the circuit size. We note that [AR23] focused on a parameter regime of constant round complexity and abritrarily small polynomial communication and verifier time (see above). Their construction also applies to our (simpler) parameter regime, see Section 3.5 and Appendix A.

Combining UOWHF trees with holographic proofs can yield useful guarantees: if a cheating prover sends a correct hash root for an encoded tableau, it can then only open the commitment to the encoding of that tableau, and the verifier can use a holographic proof system to verify complex assertions about the computation (simulating query access to the encoded tableau using openings

of the UOWHF tree). On the other hand, if the prover cheats and sends an incorrect hash root, then this is a deviation that the verifier can hope to detect (see below).

Checksums. Ideally, we would have the prover send a hash root for the (encoded) tableau of each configuration $u \in U$. However, this would require too much communication. Instead, the prover sends *checksums* for these values. We view the input U as a matrix, where each row has a configuration u. The prover computes a checksum $\widetilde{C_U}$ for this matrix and sends it to the verifier. The checksum guarantees that there is a unique matrix $\widetilde{U}$ that differs from U in at most d rows whose actual correct checksum is $\widetilde{C_U}$ (we assume there indeed exists $\widetilde{U}$ with the claimed checksum that differs from U in at most d rows, this is the more difficult case to handle). Throughout the protocol we leverage an important insight from the work of [RRR16]: once the prover sends the checksum, the matrix $\widetilde{U}$ is well defined (even though the verifier cannot directly access it). After the prover sends the checksum $\widetilde{C_U}$, the verifier chooses hash keys h for a UOWHF. The hash keys induce a hash root ρ for the encoded tableau starting at each configuration $u \in \widetilde{U}$. Since the matrix $\widetilde{U}$ was well defined before the verifier sent its hash keys, the binding guarantee of the UOWHF tree applies. Let $R(\widetilde{U})$ be the matrix of these hash roots (with dimensions $|U| \times \mathsf{poly}(\kappa)$). The prover also sends a checksum $\widetilde{C_R}$ for the matrix $R(\widetilde{U})$, and let $\widetilde{R}$ be the unique matrix that is consistent with the checksum $\widetilde{C_R}$ and differs from $R(\widetilde{U})$ in at most $2d$ rows (as above, we assume that $\widetilde{R}$ exists). If the prover is honest, then $\widetilde{U} = U$ and $\widetilde{R} = R(U)$ is the matrix of correct hash roots.

$(\widetilde{U}, \widetilde{R})$: The only game in town. At this point the prover has, in a sense, "committed" itself to the matrices $\widetilde{U}$ and $\widetilde{R}$: if the prover (responding to challenges from the verifier) makes further assertions that are not consistent with these matrices, this can be quite helpful to the verifier. In particular, suppose the protocol produces a predicate μ where the prover claims that $\mu(\widetilde{U}, \widetilde{R}) = 1$ but this assertion is false. The prover and the verifier now run an IPP to check that the prover's assertions are true for the correct input U and the matrix of correct hash roots $R(U)$: i.e. that these matrices are consistent with the checksums $\widetilde{C_U}$ and $\widetilde{C_R}$ (respectively) and that $\mu(U, R(U)) = 1$. Claim 2.1 shows that in this case $(U, R(U))$ is at least d-far from satisfying the prover's assertions. Thus, the verifier can use an IPP or IAP to get a false claim about a subset of $O(1/d)$ of $(U, R(U))$'s rows![2] We remark that the claim produced by the IPP is also about $R(U)$: this can be expressed as a low-complexity predicate that takes as input the tableaux of a subset of U's rows. The verifier's goal in the honing protocol is exactly producing such a (false) predicate. Moreover, this does not require any recursive calls to a honing protocol (beyond calls used to obtain the false assertions about $\widetilde{U}, \widetilde{R}$), and achieves excellent honing: the communication complexity is quasi-linear in d! Thus, it is in the interest of a cheating prover to avoid producing false claims about $(\widetilde{U}, \widetilde{R})$.

Claim 2.1 (Informal)**.** *The matrix $(U, R(U))$ differs in at least d rows from any matrix (A, B) that is consistent with the checksums and satisfies $\mu(A, B) = 1$.*

Proof. $\widetilde{U}$ is the unique matrix with checksum $\widetilde{C_U}$ that differs from U in at most d rows. This means that $R(U)$ and $R(\widetilde{U})$ differ in at most d rows. The matrix $\widetilde{R}$ is the unique matrix of roots that

[2]The sharp-eyed reader may notice that we need to switch from distance in rows to fractional distance of the entire matrix. In the full protocol the rows of the matrices always are encoded using a high-distance error correcting code (and this is checked by any IPP we run), so these two distances are within a constant multiplicative factor of each other. For the sake of readability, we ignore this detail in the overview.

differs from $R(\widetilde{U})$ in at most $2d$ rows and is consistent with $\widetilde{C_R}$. By a triangle inequality, $\widetilde{R}$ is the unique matrix of roots with checksum $\widetilde{C_R}$ that differs from $R(U)$ in at most d rows.

If $\mu(\widetilde{U}, \widetilde{R}) = 0$, then for every matrix (A, B) that differs from $(U, R(U))$ in at most d rows we have either: (i) $(A, B) \neq (\widetilde{U}, \widetilde{R})$, and so (A, B) is not consistent with the checksums, or (ii) $(A, B) = (\widetilde{U}, \widetilde{R})$ and thus $\mu(A, B) = 0$. □

A case analysis. From this point, the verifier's goal in the protocol is forcing the prover to make false assertions about $(\widetilde{U}, \widetilde{R})$, or to make other false statements that can be caught using a recursive call to the recursive honing protocol.

The protocol pivots on a case analysis on the number of rows in which the prover "committed" to a false root w.r.t the configurations in $\widetilde{U}$: the number of rows i in $\widetilde{R}$ that do not contain the correct root of the encoded tableaux for the i-th state in $\widetilde{U}$. We call such a row *"corrupted"*, and the analysis diverges depending on the number of corrupted rows.

2.1.1 Case I: Many Corrupted Rows

Suppose that more than k rows in $\widetilde{U}$ are corrupted (i.e., the corresponding root in $\widetilde{R}$ is not correct). In this case, the verifier chooses a uniformly random subset $J \subseteq [|U|]$ of $(|U|/k)$ rows. By a birthday-paradox-type argument, with good probability, the sub-matrix $(\widetilde{U}, \widetilde{R})|_J$ still contains a corrupted row. The prover responds with new checksums $\widetilde{C_U}'$ (with distance d) and $\widetilde{C_R}'$ for the smaller matrices $U|_J$ and $R|_J$. As explained above, if $\widetilde{C_U}'$ is not the correct checksum for $\widetilde{U}_J$ or if $\widetilde{C_R}'$ is not the correct checksum for $\widetilde{R}_J$, then the verifier will catch this at a later point (using an IPP).

Consider now the predicate ψ_{root} that gets a list of encoded tableaux (of length $(k \cdot \ell)$) induced by $|J|$ initial configurations, computes the hash root for each encoded tableau, checks that: (i) the initial configurations are consistent with the checksum $\widetilde{C_U}'$, (ii) the resulting *correct roots* are consistent with the checksum $\widetilde{C_R}'$ (and also that each encoded tableaux is correct w.r.t. its initial state). If there is indeed at least one corrupted root in $\widetilde{R}|_J$, then the encoded tableaux induced by the states in $\widetilde{U}|_J$ do not satisfy the predicate ψ_{root}. We argue that this implies that the actual input $U|_J$ (restricted to the rows in J) is at absolute distance d from any table inducing a tableau that satisfies ψ_{root}.

The verifier initiates a recursive call to the honing protocol on the smaller instance $U|_J$ with the predicate ψ_{root}. Following the naive recursive strategy described above, this call splits each of the $|J|$ computations of length $(k \cdot \ell)$ into k sub-computations of length ℓ, creating an input with $(k \cdot |J|)$ initial states for the recursive honing protocol. As established before, this leads to a k-fold degradation in the honing parameter. However, since we started this step by restricting the instance from $|U|$ to $|J| = (|U|/k)$ rows, we can afford this degradation. If the recursive honing protocol reduces the number of states by a factor of η, the final number of surviving states from the original set U will be $(\eta|U|)$. We emphasize that it was crucial that we could argue that the input to the recursive call—the sub-instance defined by J—was *far* from any table whose induced tableaux satisfy the predicate ψ_{root} (this promise is needed for the recursive honing protocol call).

2.1.2 Case II: Few Corruptions

Suppose that there are at most k rows where the root in $\widetilde{R}$ is incorrect for the corresponding configuration in $\widetilde{U}$. This means that for all but a few of configurations in $\widetilde{U}$, the corresponding

UOWHF tree root binds the prover to that configuration's correct computation tableau. In this case, we use a recursive honing protocol call as described in the naive approach: this results in a predicate $\psi_{recurse}$ and a subset $I \subset [|U|]$ of $(k \cdot \eta)$-fractional weight, where $\psi_{recurse}$ is false on (the tableaux induced by) $U|_I$. As in the naive construction, this blow-up is too large to be helpful on its own. Here, however, the recursive call serves a different purpose: taking $(k \cdot \eta) \ll 1/k$, since there were few corrupted rows to begin with, now with good probability, the roots for all the rows in I are correct. These correct roots bind the cheating prover to the encoded tableaux of $\widetilde{U}|_I$.

Now we can imagine having implicit access to (an encoding of) the tableaux induced by the states $\widetilde{U}|_I$. The prover and the verifier run a holographic proof on this huge implicit input, showing that $\psi_{recurse}$ accepts the tableaux induced by $\widetilde{U}|_I$. By the above, the holographic proof should reject w.h.p. However, verifying the holographic proof requires accessing an encoding of the huge implicit input. The verifier does not have direct access to these encodings. Still, we can build on this idea: one fact we leverage is that the verifier's check in the holographic proof can be performed using a single location v in the encoded tableau of each row in $\widetilde{U}|_I$ (the same location for all rows).

Consider the matrix M with $|U|$ rows. For each $i \in [|U|]$, the i-th row of M contains the initial configuration U_i, the root $\widetilde{R}_i$, and the correct opening of the v-th location in the encoded tableau of the computation starting at $\widetilde{U}_i$ (the opening is with respect to the hash root $\widetilde{R}_i$). Let ψ_{open} be the predicate that checks, for a given matrix of openings as above, that: (i) each opening is consistent with its corresponding root, (ii) the initial configurations and the opened values of the rows in I (one per row) would make the holographic proof verifier accept, and (iii) the initial configurations and roots are consistent with the checksums $\widetilde{C_U}$ and $\widetilde{C_R}$. We would like to argue that the "real" matrix M is at absolute distance $\Theta(d)$ from satisfying this predicate. This is almost true: there might be false openings that would make the holographic verifier accept, but they are computationally hard to find (due to the UOWHF binding guarantee). This computational distance is sufficient for our purposes: we show running an IPP (or IAP) on the matrix M with the predicate ψ_{open} allows the verifier to hone in on a small subset B_{IPP} of rows and to obtain a false claim about the tableau induced by the states in $U|_{B_{\mathsf{IPP}}}$. Indeed, this allows the verifier to reduce the set of surviving rows by a $\Theta(1/d)$ multiplicative factor (the communication and verification time of the protocol only grow polynomially with d). Note that here we didn't use the honing property of the recursive call to reduce the size of the output set, but rather to "catch" the prover in a lie about a set of rows where its checksums induce binding commitments.

2.1.3 Handling Both Cases

Error probabilities. Taking q to be the number of corrupted rows, there are two events that help the verifier make progress towards catching a cheating prover:

- In the "many corruptions" case, we need the random subset J of $|U|/k$ rows to contain a corrupted row. We take $q = \Theta(k \log\log(T))$, which means the probability of failure (i.e. the probability that there is no corrupted row) is bounded by $1/(10\log(T))$. When this failure event does not occur, the soundness guarantee of the recursive call kicks in: w.h.p. the verifier makes progress towards honing in on a smaller set and a corresponding false claim. The failure probability of $1/(10\log(T))$ is sufficiently small for our purposes, as the depth of the recursion is $o(\log(T))$. We remark that in the full construction, for succinctness, the choice of the random set J is q-wise independent, which suffices for the above analysis.

- In the "few corruptions" case, we need the recursive call (on the entire set of states) to hone in on a set (of fractional size $(k \cdot \eta)$) that does not contain any corrupted rows. Assuming that the rows chosen by the recursive call are 1-wise uniform, by a union bound this happens with probability at least $1-(k \cdot \eta \cdot q)$. For $q = \Theta(k \cdot \log\log(T))$, assuming $\eta = O(1/(k^2 \log(T) \log\log(T)))$, the failure probability here is smaller than $1/(10\log(T))$. When the failure event does not occur w.h.p. the verifier makes a lot of progress: the subsequent IPP reduces the size of the set by a $\Theta(1/d)$ factor w.h.p. As above, the failure probability is small enough to compose nicely over all levels of the recursion.

From two recursive calls to one. We argued (loosely) that each case can be handled using a single recursive honing protocol call. However, we cannot afford for the recursion to make two separate recursive calls (the depth of the recursion is $\log_k T$, and the cost would end up being more than $\mathsf{polylog}(T)$). Note, however, that the two calls are independent (the input to one call does not depend on the other). We extend the functionality of the honing protocol so that it takes as input a sequence of subsets of the entire collection of states, and a sequence of corresponding input predicates (one per subset). It returns a sequence of smaller subsets and corresponding output predicates. The soundness guarantee is that w.h.p. if there was an input set of size s rejected by its corresponding input predicate, then there will be an output set of size roughly $(\eta \cdot s)$ rejected by its corresponding output predicate. The recursion now makes a single call to a protocol with this extended functionality.

The full-blown recursive construction needs to itself provide this extended functionality, i.e. to work for a given sequence of subsets and predicates. This is achieved in a similar manner, where one crucial point is that the latter of the two recursive calls described above considers a significantly smaller set of states (of fractional size $(O(1/d))$. The complexity of the protocol grows polynomially with the number of input subsets. See Section 3.7 for a formal definition of this extended functionality, and see Section 4 for the full construction.

2.2 Improvements to Unconditional DEIPs

We can also use our techniques to construct an improved *unconditionally sound* DEIP for bounded-space computations. This result follows from an unconditionally sound recursive honing protocol (by a similar argument to the one used to construct our argument system from a computationally sound honing protocol). The main disadvantage of the unconditionally sound construction is that we make *two* recursive calls in each level of the recursion, which leads to a much less efficient proof system (albeit without using any assumptions or cryptographic machinery). To properly balance parameters, here we focus on the parameter regime where the absolute distance is $\delta_{\mathsf{ABS}} = \widetilde{O}(2^{\sqrt{\log T}})$ and the honing parameter is $\eta = 1/2^{\sqrt{\log T}}$.

Base of the recursion, length ℓ^{base}. We use an unconditionally sound IPP at the base of the recursion. In particular, we use the IPP from [RR20]: so long as the input predicate has $O(\ell^{\mathsf{base}})$ depth we get a honing protocol with honing parameter $\eta = O(1/\delta_{\mathsf{ABS}})$ and $\widetilde{O}((S \cdot \delta_{\mathsf{ABS}} + \ell^{\mathsf{base}}) \cdot \mathsf{polylog}(T))$ communication (as before, we assume $|U| \leq \mathsf{poly}(T)$).

Redux of the naive recursion. Recall the naive recursion described in Section 2.1. Starting with a honing protocol for computations of length ℓ, the idea was to construct a protocol for length

$(k \cdot \ell)$ by splitting each computation of length $(k \cdot \ell)$ into k consecutive computations of length ℓ. Taking U to be the table of states and ϕ to be the predicate that takes as input $|U|$ tableaux of length $(k \cdot \ell)$, the naive recursion uses a single call to a honing protocol for length-ℓ computations on a larger table U' and a predicate ϕ' that takes as input $|U'|$ tableaux of length ℓ. We use the fact that the table U' is comprised of $|U|$ blocks, where each block contains k mid-points of a $(k \cdot \ell)$-length computations induced by the single state in U that corresponds to that block.

As before, the recursive call outputs a subset B' of the states in U' with fractional size η, and a predicate ψ' on the length-ℓ tableaux of the U'-states in this smaller subset. Going back to the original input U, we can take B to be the subset containing each state in U whose corresponding block in U' contains at least one state that "survived" in B'. Similarly, we can take ψ to be the predicate over the $(k \cdot \ell)$-length tableaux of states in B that checks that ψ' accepts the length-ℓ sub-tableaux corresponding to states that survived in B'. This naive recursion is sound: if the tableaux of U were far from satisfying ϕ, then the tableaux of the surviving states in B' should not satisfy ψ. The issue (as before) is that a $(k \cdot \eta)$-fraction of the states in U "survive" in B, and this is too large of a degradation in the honing parameter.

Adding a second recursive call. To further improve the homing parameter, we make a second recursive call, checking that ψ accepts the tableaux of the states in B. As with the first call, we "expand" each state u in $U|_B$ into k states in a new table $(U|_B)'$, corresponding to the k mid-points of the $(k \cdot \ell)$-length computation starting at u, and also define a new predicate ϕ_2 over $k \cdot |B|$ tableaux of length ℓ, which checks internal consistency and that ψ accepts. Following the above logic, we'd get a new predicate ψ_2 over a $(k \cdot \eta)$-fraction of the states in B. Overall, only a $(k \cdot \eta)^2 \ll \eta$-fraction of the states in U survive, so the honing parameter is maintained (we can assume that $\eta < 1/k^2$ and guarantee this throughout the recursion). The remaining issue is that the second recursive call is only guaranteed to be sound if its input is *far* from satisfying the predicate, and we have no distance guarantee: all we know is that the tableau of the states in B fail to satisfy ψ, so the tableaux of the states in the expansion of $U|_B$ fail to satisfy ϕ_2, but there is no guarantee that they are *far* from satisfying it.

Ensuring distance. Before the first recursive call, the prover sends a checksum $\widetilde{C_U}$ for the table of states U (as in the construction of Section 2.1). We show that after the recursive call w.h.p. U is far from any table V that both: (i) has checksum $\widetilde{C_U}$, and (ii) the $(k \cdot \ell)$-tableaux of the states $V|_B$ satisfy the predicate ψ. Importantly, the first of these two conditions only looks at the honest-to-God states in U, rather than their tableaux. We can now look at an expanded table U'', where we only expand the rows of states in B. Namely, U'' includes each state in U, and for the states in B it also includes the block of k intermediate states of that state's $(k \cdot \ell)$-length computation. By the above, the length-ℓ tableau of U'' are far from satisfying the predicate ϕ'' that checks that (i) the first states in the tableaux of the states that are in U have checksum $\widetilde{C_U}$, and (ii) the tableau of blocks corresponding to states in B are internally consistent (each block of k tableaux of length ℓ corresponds to a correct computation of length $(k \cdot \ell)$) and they satisfy the predicate ψ. Now, since we have distance, the recursive call should return a subset B'' of an η-fraction of the rows of U'' and a predicate ψ'' over the length-ℓ tableaux of those rows. We translate this into a predicate ψ_2 over the $(k \cdot \ell)$-length tableaux of the rows in a subset B_2 of U's rows. The subset B_2 includes: (a) the rows of U that were not in B (and thus were not expanded in U'') but survived in B'', and (b) the rows of U in the set B for which at least one of their k midpoints survived in B''. By the

guarantee of the recursive call, an η fraction of U' states that were outside B survive, and a $(k \cdot \eta)$ fraction of the states in B. Since B itself was of fractional weight $(k \cdot \eta)$ in U, we conclude that the fraction of states in U that survive the second recursive call is:

$$(1 + k^2 \cdot \eta) \cdot \eta \leq (1 + 1/k) \cdot \eta,$$

where the inequality holds assuming that $\eta \leq 1/k^3$. We can guarantee this at the base of the recursion and maintain it throughout the recursion (the depth of the recursion is less than k, so the mild growth in η stays under control).

Cost of the recursion. Other than adding a second recursive call (which doubles the cost of the protocol), the main additional cost is the communication for sending the checksum: we need the honing parameter to satisfy $\eta \leq 1/k^3$ (see above). We always need the absolute distance to be at least $1/\eta$, so this requires $\delta_{\mathsf{ABS}} = \Omega(1/\eta)$. We pay for this absolute distance in the checksum, which needs to be of length $\widetilde{O}(\delta_{\mathsf{ABS}} \cdot S)$ (to guarantee the aforementioned distance). Thus, the recursion incurs a $\widetilde{O}(k^3 \cdot S)$ additive communication and verifier runtime overhead. We remark that the polynomial dependence on k can be improved (in this construction we can allow a constant factor degradation in the honing parameter from each recursive call), but this would not improve the asymptotic complexity of the resulting protocol.

The resulting honing protocol. Each recursive step roughly doubles the cost of the protocol, while increasing the length of the computations by a multiplicative factor of k. The recursive calls are expensive, so we want to take k to be as large as possible, but we pay for this with an additive $\widetilde{O}(k^3 \cdot S)$ term in the communication and verifier runtime. In the base of the recursion, we start with a base protocol for length-k computations that has communication complexity $\widetilde{O}(k \cdot S)$. After r levels of recursion we get a protocol for computations of length $k^{r+1} = T$. The communication complexity is $(2^r \cdot \mathsf{poly}(k) \cdot S)$. To balance the parameters we want $2^r \approx \mathsf{poly}(k)$, which is achieved (asymptotically) by taking $k = 2^{\sqrt{\log T}}$ and $r = O(\sqrt{\log T})$. The resulting honing protocol for computations of length T has $(2^{O(\sqrt{\log T})} \cdot S)$ communication and verifier runtime. The improved DEIP of Theorem 5.5 follows from this honing protocol similarly to the template used above for argument systems.

3 Preliminaries and Definitions

For a string $x \in \Sigma^n$ and an index $i \in [n]$, we denote by $x_i \in \Sigma$ the i^{th} entry in x. If $I \subseteq [n]$ is a set then we denote by $x|_I$ the sequence of entries in x corresponding to coordinates in I.

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 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, take its minimum distance to be the minimum, over all distinct vectors $x, y \in S$ of $\Delta(x, y)$. We use $\Delta(S)$ to denote the minimum distance of S. Fixing a vector space, for a set S and a vector x, we denote $(x + S) = \{x + y : y \in S\}$. For a scalar c, we denote $(c \cdot S) = \{c \cdot y : y \in S\}$.

Nested sets. We use sequences of sets $(A_i \subseteq U)_{i\in[m]}$ that are *nested* (where m is a positive integer):

Definition 3.1 (Nested sets)**.** *We say that a sequence of subsets $(A_i \subseteq U)_{i\in[m]}$ over a universe U are* nested *if $A_m \subseteq A_{m-1} \subseteq \ldots A_2 \subseteq A_1$. By convention, the largest subset is the first and the subsets shrink as i increases.*

Disjoint union. For two sets A and B, the disjoint union $C = A \coprod B$ includes each element that appears in A or in B. Similarly to a multi-set, elements that appear in both sets appear twice in C. However, the disjoint union indexes elements to identify their origin:

Definition 3.2 (Disjoint Union)**.** *The disjoint union of A and B is:*

$$A \coprod B = \{(a, 0) : a \in A\} \cup \{(b, 1) : b \in B\}.$$

We use disjoint unions such as $C = A \coprod B$ to index the rows of matrices. For such a matrix M with $|C| = |A| + |B|$ rows, we use C to refer to the row-set of the matrix. We abuse notation and index the first $|A|$ rows of M using the elements of A (implicitly referring to $A \times \{0\}$) and the last $|B|$ rows using elements of B (implicitly referring to $B \times \{1\}$). This is an abuse of notation because the same element can appear in both sets, but we find that it simplifies the notation. The "origin" of each element we refer to (from A or from B) will always be clear from the context.

3.1 Cryptographic Security, UOWHFs and Hash Trees

We focus on security against non-uniform adversaries. We parameterize security using a security parameter κ and functions $\lambda : \mathbb{N} \to \mathbb{N}, \varepsilon : \mathbb{N} \to (0, 1)$, we say that a cryptographic object is (λ, ε)-secure if the success probability of any adversary of size $\lambda(\kappa)$ is bounded by $\varepsilon(\kappa)$. For example, the standard notion of one-way functions requires security (hardness to invert) against any polynomial size function λ and any inverse-polynomial success probability ε. Subexponential one-way functions require that there exists a constant $\sigma > 0$ where security holds for any size bound of the form $O(\mathsf{exp}(\sigma \cdot \kappa))$ and success probability of the form $\Omega(\mathsf{exp}(-\sigma \cdot \kappa))$.

Throughout this work, when we say that a construction is $(\lambda(\kappa)/\mathsf{poly}(\kappa), \varepsilon(\kappa) \cdot \mathsf{poly}(\kappa))$-secure, we mean there exists a *fixed* polynomial function f s.t. it is $(\lambda(\kappa)/f(\kappa), \varepsilon(\kappa) \cdot f(\kappa))$-secure

Definition 3.3 (UOWHF [NY89])**.** *Let κ be a security parameter, let $\lambda : \mathbb{N} \to \mathbb{N}, \varepsilon : \mathbb{N} \to [0, 1]$ be functions controlling security 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 $h \in \mathcal{H}_\kappa, h : \{0, 1\}^{n_{1_\kappa}} \to \{0, 1\}^{n_{0_\kappa}}$.*

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

1. *If $x^{(1)} \in \{0, 1\}^{n_{1_k}}$ is A's initial value, then $\Pr[A(h, x^{(1)}) = x^{(2)}, h(x^{(1)}) = h(x^{(2)}), x^{(2)} \neq x^{(1)}] \leq \varepsilon(\kappa)$, where the probability is taken over $h \in \mathcal{H}$ and the random choices of A.*

2. *$\forall h \in \mathcal{H}$ there is a description of h of length polynomial in n_{1_κ}, 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^κ, 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.*

Rompel gave the first construction of UOWHFs from arbitrary one-way functions. Katz and Koo gave a full proof for Rompel's construction.

Theorem 3.4 ([Rom90, KK05])**.** *The existence of (λ, ε)-secure one-way functions implies the existence of $(\lambda(\kappa)/\mathsf{poly}(\kappa), \varepsilon(\kappa) \cdot \mathsf{poly}(\kappa))$-secure* UOWHF*s.*

We follow the hash tree construction and definition from the work of Amit and Rothblum [AR23]. They define a "relaxed" (when comparing to CRHs) security property that a commitment scheme based on a Merkle tree may satisfy, and call such a commitment a "2-PLOSC" (Definition 3.7). They show in Claim 3.8 that instantiating a Merkle tree with a family of UOWHFs results in a 2-PLOSC. We emphasize 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 3.5 (Hash Tree [AR23])**.** *Fix $N, n_{in}, n_{out} \in \mathbb{N}$ and a finite alphabet Σ. Set*[3] *$L = \frac{N}{n_{in}}$, 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:*

- *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 $w_1 = y$.*

Following [AR23], we index "chunks" of n_{in} nodes, which they call a *block*: for each fixed tree layer i, the blocks in this layer are indexed as $c_{(i,1)}, \ldots, c_{(i,M)}$ for $M = |w_i|/n_{in}$.

Definition 3.6 (Valid Path)**.** *Let $N, n_{in}, n_{out}, \Sigma, L, \ell$ and $h_1, ..., h_\ell$ be as in Construction 3.5. 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 3.7 (2-PLOSC [AR23])**.** *Let $\lambda : \mathbb{N} \to \mathbb{N}, \varepsilon : \mathbb{N} \to (0, 1)$, let $N, n_{in}, n_{out}, \Sigma, \ell$ and $h_1, ..., h_\ell$ be as in Construction 3.5 and parameterized by a security parameter κ. Assume that $n_{out} \ll N$. A* Second-Preimage Locally Openable Succinct 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:*

[3] We assume that N, as well as any other layer length, is a multiple of n_{in}. This is w.l.o.g. by a padding argument.

- **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 every string* $z \in \Sigma^N$*, for every* $\lambda(\kappa)$*-size adversary* $\mathcal{A}$ *that follows the protocol in the commit phase (sending the correct* y*) and then later tries to cheat in the local opening phase by outputting* $q^* \in [N]$ *and a path* p^**, taking* $q' = \left\lceil \frac{q^*}{n_{in}} \right\rceil$*:*

$$\Pr_{\substack{\mathcal{A}\text{'s coins} \\ h_1,...,h_\ell}} [p^* \text{ is a valid path from } q' \text{ to } y \text{ and } p^*_{\ell+1} \neq z[(q'-1)n_{in}]...z[q' \cdot n_{in} - 1]] \leq \varepsilon(\kappa).$$

Claim 3.8 (A UOWHF tree is a 2-PLOSC [AR23])**.** *In the setting of Definition 3.7, if* R *samples* $h_1, ..., h_\ell$ *uniformly at random from a* (κ, ε) *UOWHFs family* $\mathcal{H} : \Sigma^{n_{in}} \to \Sigma^{n_{out}}$*, and* n_{in} *and* N *are polynomially related, then the commitment scheme is* (λ', ε') *secure, where* $\lambda' = \lambda(\kappa)/\mathsf{poly}(\kappa, N, \log|\Sigma|)$ *and* $\varepsilon' = \varepsilon(\kappa) \cdot \mathsf{poly}(\kappa, N, \log|\Sigma|)$*.*

3.2 Succinct Descriptions

Throughout this work we use NC to refer to the class of logspace uniform Boolean circuits of polylogarithmic depth and constant fan-in. Namely, $\mathcal{L} \in \mathsf{NC}$ if there exists a logspace Turing machine M that on input 1^n outputs a full description of a polylogarithmic depth circuit $C : \{0,1\}^n \to \{0,1\}$ such that for every $x \in \{0,1\}^n$ it holds that $C(x) = 1$ if and only if $x \in \mathcal{L}$. NC^1 is the subclass of NC where the depth is logarithmic.

Following [RR20], we use succinct representations of functions and sets. Loosely speaking, a function $f : \{0,1\}^n \to \{0,1\}$ has a succinct representation if there is a short string $\langle f \rangle$ that describes f. That is, $\langle f \rangle$ can be expanded to a full description of f. The actual technical definition is slightly more involved and in particular requires that the full description of f be an NC (i.e., polylogarithmic depth) circuit:

Definition 3.9 (Succinct Description of Functions)**.** *We say that a function* $f : \{0,1\}^n \to \{0,1\}$ *of size* $s(n)$ *has a* $\rho(n)$-succinct description *if there exists a string* $\langle f \rangle$ *of length* $\rho(n)$ *and a logspace Turing machine* M *(of constant size, independent of* n*) such that on input* 1^n*, the machine* M *outputs a full description of an* NC *circuit* C *such that for every* $x \in \{0,1\}^n$ *it holds that* $C(\langle f \rangle, x) = f(x)$*. We refer to* $\langle f \rangle$ *as the succinct description of* f*. If* $\rho(n) = \mathsf{polylog}(n)$ *then we say that the function is* fully succinct*.*

We also define succinct representation for sets $S \subseteq [k]$. Roughly speaking this means that the set can be described by a short string. The formal definition is somewhat more involved:

Definition 3.10 (Succinct Description of Sets)**.** *We say that a set $S \subseteq [k]$ of size $s(k)$ has a $\rho(k)$-succinct description if there exists a string $\langle S \rangle$ of length $\rho(k)$ and a logspace Turing machine M such that on input 1^k, the machine M outputs a full description of a depth $\mathsf{polylog}(k)$ and size $\mathsf{poly}(s, \log k)$ circuit (of constant fan-in) that on input $\langle S \rangle$ outputs all the elements of S as a list (of length $s \cdot \log(k)$). If $\rho(k) = \mathsf{polylog}(k)$ then we say that the set is* fully succinct.

We emphasize that the size of the circuit that M outputs is proportional to the actual size of the set S, rather than the universe size k.

3.3 Reversible Computation, Tableau and Multi-Tableau

For simplicity, we focus on single-tape Turing Machines (TMs). A configuration of a Turing machine $\mathcal{M}$ can be described by a string that includes the contents of the work tape, the head positions on the input and the work tape, and the current state (extending this to multi-tape TMs is straightforward and would not meaningfully change any of our results). Note that we do not include the input x in the description of a configuration. If the machine has space bound $S \geq \log(|x|)$, then the configuration can be described using $O(S)$ bits. The machine has a starting configuration u_{start} and w.l.o.g we assume it has a unique accepting configuration u_{accept}.

Reversible computation. In a reversible Turing Machine, the transition function comes with an inverse, so the computation can be run forward or backward. We capture this by assuming that the Turing Machine has a step / successor function succ (as usual) and also a predecessor function pred. These functions take as input a configuration u for the machine (formally, these functions also depends on the input x: we assume x is fixed and ignore it in the notation for now). For every configuration u on the computation path starting at u_{start}: $\mathsf{pred}(\mathsf{succ}(u)) = u$. Note that for configurations that are not on the path starting at u_{start} this equality might not hold.

Bennett [Ben89] studied reversible computation and showed that any Turing Machine running in time T and space S could be simulated by a *reversible* Turing Machine running in time $\mathsf{poly}(T)$ and space $O(S \cdot \log(T))$. In this work, we assume all Turing Machines are reversible. This incurs an overhead that does not meaningfully change any of our results.

The main advantage in using reversible TMs is that if we look at two computations that start at different states u and u' and run for the same number of steps (and on the same input): either at some point one of the computations reaches a state where the successor and predecessor functions disagree (and this means it isn't on the correct computation path), or the configurations traversed by the two computations disagree at every point on the two paths. See Fact 3.12 below.

Computation tableau (reversible computation). Fixing an input x, an initial state u and a computation length ℓ, the tableau of $\mathcal{M}$'s computation on x starting in state u is a sequence of ℓ such configurations, representing the evolution of the machine's state over each computation step (we consider tableaux beginning at general states u, not only the machine's specified start state u_{start}). If the computation terminates before ℓ steps, the remaining configurations are set to a default "blank" symbol. If at any point in the computation, the successor and predecessor disagree then the remaining configurations are set to a special "corrupted" symbol $\perp$.

Definition 3.11 (Computation tableau, multi-tableau)**.** *Let $\mathcal{M}$ be a Turing machine, let x be an input, let $S \geq \log(|x|)$ be a bound on the space used by $\mathcal{M}$ on inputs of length $|x|$ and let ℓ be a*

computation length (we usually assume that ℓ is smaller than the worst-case runtime of $\mathcal{M}$, starting at u_{start}, on inputs of length $|x|$). The computation tableau $\mathcal{T}_{u,x,S,\ell}$ *is a sequence of configurations $u_0, u_1, \ldots, u_\ell$, where:*

- $u_0 = u$.
- *For $i \leftarrow 1, \ldots, \ell$:*

  if $u_{i-1} = \mathsf{pred}(\mathsf{succ}(u_{i-1}))$, then $u_i = \mathsf{succ}(u_{i-1})$ (both the successor and predecessor functions are computed with x on the input tape).

  Otherwise, $u_i = \perp$, and we define $\mathsf{succ}(\perp) = \mathsf{pred}(\perp) = \perp$, so also for every $j \in [i+1, \ell]$ $u_j = \perp$

The tableau can thus be represented as a string of length $O(\ell \cdot S)$. We sometimes omit the subscripts x, u, ℓ, S (or subsets thereof) when they are clear from the context.

For a tuple $U = (U_1, \ldots, U_N)$ of configurations, we define concatenation of the tableaux:

$$\mathcal{T}_{U,x,S,\ell} = (\mathcal{T}_{U_1,x,S,\ell}, \ldots, \mathcal{T}_{U_N,x,S,\ell}) .$$

We refer to $\mathcal{T}_{U,x,S,\ell}$ as a multi-tableau. *We sometimes view U as a matrix of dimensions $|U| \times O(S)$ (so we refer to configurations in U as "rows"). Similarly, we sometimes view $\mathcal{T}_{U,x,S,\ell}$ as a matrix of dimensions $N \times O(\ell \cdot S)$*

Fact 3.12. *For a reversible Turing Machine $\mathcal{M}$, for any two configurations u, u' and any input x, space bound S and computation length ℓ, either it is the case that $\mathcal{T}_{u,x,S,\ell}$ and $\mathcal{T}_{u',x,S,\ell}$ differ in all locations, or for at least one of them there is a location in the tableau where the successor and predecessor functions disagree: e.g. if this holds for u then $\exists j \in [0, \ell - 1]$ s.t. $u_j \neq \mathsf{pred}(\mathsf{succ}(u_j))$ and thus $\mathcal{T}_{u,x,S,\ell}|_{[j+1,\ell]} = (\perp, \ldots, \perp)$.*

Remark 3.13. *In this work, whenever we say that a sequence is a consistent (or valid) tableau for a Turing Machine, this means that none of the configurations is corrupted / has value $\perp$. In particular, for every configuration in a consistent tableau, the successor and predecessor functions must agree. By Fact 3.12, if we check that the tableaux beginning at states $u \neq u'$ are both consistent, then these tableaux disagree everywhere. Further, this holds even if only one of the tableaux is consistent: suppose u's tableau is the valid one, then the two tableau disagree on the prefix where u''s is uncorrupted, and they also disagree on the suffix where u''s is corrupted (because u's is not).*

The Multi-Tableau Language. We define a language that asserts properties of a sequence of computation tableaux. The computations are all specified w.r.t the same Turing machine $\mathcal{M}$, input x, space bound S, and computation length ℓ, but they each begin at a different initial configuration taken from a tuple U: for each configuration $u \in U$, we consider the ℓ-step tableau $\mathcal{T}_u$ starting at u (a string of length $O(\ell \cdot S)$). The language checks that the entire table of all tableaux satisfies a given (succinct) predicate.

For technical reasons (see below), rather than considering a single tuple of configurations and a single predicate (to be applied to the concatenation of the tableaux), we consider a sequence of nested subsets (see Definition 3.1) of the tuple, and a corresponding sequence of succinct predicates ϕ_i (one predicate for the tableaux of the rows in each subset).

Definition 3.14 (Multi-tableau Language)**.** *Let $\mathcal{M}$ be a Turing machine, $x \in \{0,1\}^*$ be an input, $S \in \mathbb{N}$ be a space bound and $\ell \in \mathbb{N}$ be a computation length. Let U be a tuple of configurations, and let $A_1, \ldots, A_m \subseteq [|U|]$ be a sequence of nested subsets (i.e. $A_m \subseteq \cdots \subseteq A_2 \subseteq A_1$). Let $(\phi_1, \ldots, \phi_m)$ be a corresponding sequence of succinct predicates (see Definition 3.10 and Definition 3.9).*

An instance $(\mathcal{M}, x, S, \ell, U, (A_i)_{i=1}^m, (\phi_i)_{i=1}^m)$ is in the language $\mathcal{L}_{\text{multi-tableau}}$ if for every $i \in [m]$:

$$\phi_i\left(\mathcal{T}_{U|_{A_i},x,\ell,S}\right) = 1.$$

Remark 3.15 (Single-Set Multi-Tableau)**.** *In some cases we focus on the case where there is only a single predicate ϕ that applies to the entire tableau (i.e. the full set $[|U|]$). We refer to this as the single-set case, and in this case we omit the collection of sets (A_i) from the notation and refer to a single predicate ϕ (with no subscript).*

Expanding and contracting multi-tableau. Our protocols move from considering a (tuple of) $(k \cdot \ell)$-length tableau, to k tableaux, each of length ℓ. The idea is to look at k mid-points in the $(k \cdot \ell)$-length computation, each reached from the previous one by taking ℓ computation steps. We refer to this as the *k-fold expansion* of a tableau or a multi-tableau.

Definition 3.16 (Expanding and contracting multi-tableau)**.** *Fix a Turing machine $\mathcal{M}$, an input x, a space bound S, a computation length ℓ and a parameter k. Let U be a tuple of configurations. The k-fold* expansion *of U, which we denote U^{expd}, is a tuple of $(k \cdot |U|)$ configurations. For $i \in [U], j \in [k]$, the (i,j)-th configuration in U^{expd} is the $((j-1)\cdot\ell)$-th configuration in $\mathcal{T}_{U_i,x,S,\ell}$: i.e. the $((j-1)\cdot\ell)$-th configuration reached by a computation starting at the configuration U_i.*

We alternate between indexing the rows of the k-fold expansion by elements in $[|U|] \times [k]$ or by elements in $[k \cdot |U|]$ in the natural way. The i-th block *of U is the states $(U_{(i,1)}, \ldots, U_{(i,k)})$. For a subset $A \subseteq [|U|]$, its k-fold expansion $A^{\mathsf{expd}} = A \times [k]$ is a subset of the rows of U^{expd}.*

For a set $B \subseteq [|U|] \times [k]$ of rows in the k-fold expansion, we also define its k-fold contraction *$B^{\mathsf{ctrt}} = \{i : \exists j \in [k] \text{ s.t. } (i,j) \in B\}$. Note here that B itself might not be a product set: for some indices $i \in [|U|]$ it might have 1 row or a few rows from the i-th block.*

For a matrix W with $[|U|] \times [k]$ rows, the k-fold contraction *of W, which we denote W^{ctrt} is the sub-matrix of W that only includes the first row of each block, i.e. the rows of the form $\{(i,1)\}_{i \in [|U|]}$. Alternatively: $W^{\mathsf{ctrt}} = W|_{([U]\times[k])^{\mathsf{ctrt}}}$.*

By construction it always holds that $U = (U^{\mathsf{expd}})^{\mathsf{ctrt}}$. Similarly, for a matrix W with row set $[|U|] \times [k]$, if each block i of W contains the correct mid-points of the tableau starting at configuration $W[(i,1)]$, then $W = (W^{\mathsf{ctrt}})^{\mathsf{expd}}$.

Remark 3.17. *Our recursive honing protocol construction for computations of length $(\ell \cdot k)$ recursively operates on the k-fold expansion of an input matrix U. The recursive call returns a subset B of the rows of U^{expd} and a claim about the ℓ-length tableaux of those rows. We can (and do) interpret this as a claim about the $(k \cdot \ell)$-length tableaux of the rows of U that are in B^{ctrt}: the $(k \cdot \ell)$-length tableau of a state U_i includes within it the ℓ-length tableau of every state $(U_{i,j})_{j \in [k]}$.*

3.4 Polynomials, Low Degree Extensions and Checksums

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

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

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 3.19. *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 (see, e.g., [Gol08, Appendix G.3] for details).

3.4.1 Low Degree Extension

Let $\mathbb{H}$ be a finite field and $\mathbb{F} \supseteq \mathbb{H}$ be an extension field of $\mathbb{H}$. 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:

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

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

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

The function $\hat{\phi}$ is called the *low degree extension* of ϕ (with respect to $\mathbb{F}$, $\mathbb{H}$ and m). In the special case in which $\mathbb{H} = \mathbb{GF}(2)$, the function $\hat{\phi}$ (which has individual degree 1) is called the *multilinear extension* of ϕ (with respect to $\mathbb{F}|$ and m).

A low-degree extension for multiple strings. We extend the $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ encoding to encode multiple strings (of varying lengths). A natural way of doing so is by simply concatenating the strings and applying $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}$ to the concatenated string. However, following past work and in particular [RRR16], we take a slightly different approach. This is useful both for composing encodings of individual strings to a single joint encoding, and for decomposing such a joint encoding into individual encodings. This approach leverages the fact that the low degree extension is a tensor code [Wol65].

Given strings $x_1 \in \{0,1\}^{n_1}, \ldots, x_k \in \{0,1\}^{n_k}$, we define an encoding $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_1, \ldots, x_k)$ as follows. Let $m_k = \log_{|\mathbb{H}|}(k)$, $m^{(i)} = \log_{|\mathbb{H}|}(n_i)$, $m_{\max} = \max_{i \in [k]} m^{(i)}$ and $m = m_k + m_{max}$. We

identify $[k]$ with $\mathbb{H}^{m_k}$ and identify $[\max(|x_i|)]$ with $\mathbb{H}^{m_{max}}$ by viewing the respective integers in base $|\mathbb{H}|$. Let $P : \mathbb{H}^m \to \{0,1\}$ be a function such that for every $z \in \mathbb{H}^{m_k}$, and every $(w,y) \in \mathbb{H}^{m^{(z)}} \times \mathbb{H}^{m-m_k-m^{(z)}}$, it holds that: $P(z,w,y)$ is equal to the w^{th} bit of x_z if $y = 0^{m-m_k-m^{(z)}}$, and $P(z,w,y) = 0$ otherwise (i.e., if $y \neq 0^{m-m_k-m^{(z)}}$). We define $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_1,\ldots,x_k)$ as the low degree extension of P with respect to $\mathbb{F}$, $\mathbb{H}$ and m.

Proposition 3.20. *Let $\mathbb{H}$ and $\mathbb{F}$ be constructible field ensembles such that $\mathbb{F}$ is an extension field of $\mathbb{H}$ and $|\mathbb{H}| \geq \log(|\mathbb{F}|)$. There exist procedures* Compose *and* Decompose *as follows:*

- **Compose***: Given oracle access to $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_1),\ldots,\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_k)$ and a point $z \in \mathbb{F}^m$, where m is as above, the procedure* Compose *makes a single query to each $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_i)$ (the same query to each of these* LDE*s and k queries in total), runs in time $k \cdot \mathsf{poly}(|\mathbb{H}|, \sum_i \log_{|\mathbb{H}|}(n_i), \log(k))$ and outputs the z^{th} entry of $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_1,\ldots,x_k)$.*
- **Decompose***: Given oracle access to $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_1,\ldots,x_k)$, some $i \in [k]$ and a point $w \in \mathbb{F}^{m^{(i)}}$, where $m^{(i)}$ is as above, the procedure* Decompose *makes a single oracle query, runs in time $\mathsf{poly}(|\mathbb{H}|, \log(k), \sum_i \log_{|\mathbb{H}|}(n_i))$ and outputs the w^{th} entry of $\mathsf{LDE}_{\mathbb{F},\mathbb{H}}(x_i)$.*

LDE predicates. We point out some specific scenarios where we use LDEs in this work. The first is an *LDE predicate*: a function that checks whether the LDE of its input string takes on a given value $v \in \mathbb{F}$ at a given coordinate $z \in \mathbb{F}^m$. An LDE predicate $f_{\mathbb{H},\mathbb{F},m,z,v} : \mathbb{F}^{\mathbb{H}^m} \to \{0,1\}$ outputs 1 iff, when we view its input string as a function $\phi : \mathbb{H}^m \to \mathbb{F}$, it holds that $\phi(z) = v$ (otherwise it returns 0). LDE predicates can be decomposed as per Proposition 3.20: the input is viewed as the concatenation of several sub-inputs $x = (x_1, \cdots, x_k)$, and verifying an LDE claim about the entire input can be reduced to verifying LDE sub-claims about the sub-inputs (the untrusted sends alleged values for the sub-claims to the verifier).

Encoded tableau. For given fields $\mathbb{H}, \mathbb{F}$, space bound S, initial configuration u and runtime ℓ, the encoded tableau of a machine $\mathcal{M}$ on input x is the encoding we get by interpreting the tableau $\mathcal{T}_{u,x,S,\ell}$ as a string in $\mathbb{H}^m$ for $m = \lceil \log_{|\mathbb{H}|} |\mathcal{T}_{u,x,S,\ell}| \rceil$ (if we need to round m up, we pad the suffix of the tableau with 0's). The encoding is performed by recursively diving the tableau into $|\mathbb{H}|$ sub-tableaux, each of (up to) $\lfloor \ell/|\mathbb{H}| \rfloor$ configurations, encoding the sub-tableaux and composing those encodings as in Proposition 3.20.

We typically take $|\mathbb{H}| = O(\log(T))$ and $|\mathbb{F}| = \mathsf{polylog}(T)$ where T is the total size of a computation of interest in the protocol we are constructing (this can be longer than the particular parameter ℓ we encode at any one point in the protocol).

3.4.2 Checksum

We use a checksum construction from [RRR16]. Let $k \in \mathbb{N}$ and $d \leq \frac{k}{\log(k)}$, we use a *systematic*[4] linear error-correcting code $C : \mathbb{F}^k \to \mathbb{F}^{k+2d\cdot\log(k)}$ with absolute distance $2d+1$ and $\mathsf{poly}(\log(|\mathbb{F}|), k)$ time encoding and decoding. We use this code as a *d-checksum* function, where for any fixed checksum value $chksum \in \mathbb{F}^{2d\log(k)}$, and any vector $x \in \mathbb{F}^k$, there is at most one vector $x' \in \mathbb{F}^k$ with checksum $chksum$ at absolute distance d from x. Moreover, x' can be recovered from $(x, chksum)$ in quasi-linear time.

[4]Recall that an error-correcting code C is systematic if there exists a function ENC such that $C(\eta) \equiv (\eta, \mathsf{ENC}(\eta))$.

The parameters are captured in the following proposition from [RRR16], where we take d to be the blowup in the checksum, which is linear in the unique decoding radius.

Proposition 3.21 ([RRR16])**.** *Let $\mathbb{F}$ be a constructible field ensemble (see Definition 3.19) and let $k = k(n) \geq 1$ and $d = d(n) \leq k/\log(k)$. There exists a* systematic *$\mathbb{F}$-linear error-correcting code $C : \mathbb{F}^k \to \mathbb{F}^{k+d\cdot\log(k)}$ with absolute distance $d+1$ and $k \cdot \mathsf{poly}(\log(|\mathbb{F}|), \log(k))$-time encoding and decoding. For fields of (at most) $\mathsf{poly}(n)$ size, the encoding function is in log-space uniform* NC.

The classical Reed-Solomon code almost satisfies the requirements in Proposition 3.21 (even with slightly better rate): it requires the field size to be quite large (i.e., $|\mathbb{F}| \gg k$), which can be inconvenient (unlike [RRR18], this is not a crucial issue for us, but it is still more convenient to avoid a dependence of $|\mathbb{F}|$ on k). The construction of [RRR16] uses a Reed-Solomon code, but over a (sufficiently large) extension field.

Since C is systematic, there exists a function $\mathsf{ENC} : \mathbb{F}^k \to \mathbb{F}^{d\cdot\log(k)}$ such that $C(\eta) \equiv (\eta, \mathsf{ENC}(\eta))$. We will often refer to $chksum = \mathsf{ENC}(\eta)$ as the checksum of the vector η.

The function $\mathsf{DEC} : \mathbb{F}^{k+d\cdot\log(k)} \to \mathbb{F}^k$ is the decoding function for C that *preserves checksums*. That is, on input $\eta \in \mathbb{F}^k$ and $chksum \in \mathbb{F}^{d\log(k)}$, the function DEC outputs a vector $\eta' \in \mathbb{F}^k$ at distance $\lfloor (d+1)/2 \rfloor$ from η with $\mathsf{ENC}(\eta') = chksum$ if such an η' exists, or $\perp$ otherwise. We remark that the regular notion of unique decoding suffices here: if such an η' exists, then it is the unique codeword of C that is $\lfloor (d+1)/2 \rfloor$-close to $C(\eta)$. I.e., the checksum decoder DEC runs the unique decoding algorithm of C. If the unique decoder returns a $\lfloor (d+1)/2 \rfloor$-close codeword with checksum $chksum$ then DEC can simply output (the prefix of) that codeword. Otherwise, if the unique decoding algorithm fails or returns a closest codeword with checksum different from $chksum$, then DEC can safely return $\perp$ (there is no close codeword with checksum $chksum$).

We extend ENC and DEC to operate over matrices (rather than just vectors) by operating row-by-row. That is, $\mathsf{ENC} : \mathbb{F}^{m\times k} \to \mathbb{F}^{m\times(d\cdot\log(k))}$ is defined as $\mathsf{ENC}(\eta_1, \ldots, \eta_m) \equiv \big(\mathsf{ENC}(\eta_1), \ldots, \mathsf{ENC}(\eta_m)\big)$ (where the η_i's are the rows of the matrix). Similarly, we define $\mathsf{DEC} : \mathbb{F}^{m\times k} \times \mathbb{F}^{m\times d\cdot\log(k)} \to \mathbb{F}^{k\times m}$:

$$\mathsf{DEC}\Big((\eta_1, chksum_1), \ldots, (\eta_m, chksum_m))\Big) \equiv \Big(\mathsf{DEC}(\eta_1, chksum_1), \ldots, \mathsf{DEC}(\eta_m, chksum_m)\Big).$$

3.5 Holographic Interactive Proofs and Arguments

In a *Holographic* Interactive Proof (HIP) the verifier, rather than having explicit access to its input, has query access to an *encoding* of its input. The hope is that the redundancy provided by the encoding will allow for improving the efficiency of the proof-system (in particular, allowing the verifier to run in sublinear or even polylogarithmic time). Most commonly, the encoding is a low-degree extension, as originally proposed by [BFLS91] (in the PCP context). Holographic proofs with general codes were formalized by Gur and Rothblum [GR17].

We focus on LDE encodings, and on protocols where the verifier never accesses its holographic input during the interaction: at the end of the protocol, it outputs a claim about the value of the encoding at a single coordinate, i.e. it outputs an LDE predicate (see Section 3.4.1).

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

Definition 3.22 (Holographic Interactive Proof (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 Proof *for a pair language $\mathcal{L}$, with*

respect to the low degree extension LDE*, is an interactive protocol between a prover* $\mathcal{P}$ *and a verifier* $\mathcal{V}$*. Both parties get as input* $x \in \{0,1\}^{n_{\mathsf{exp}}}$*. The prover also gets* $w \in \{0,1\}^{n_{\mathsf{hol}}}$ *whereas the verifier only gets* $|w| = n_{\mathsf{hol}}$*. We take* $m = \lceil \log_{|H|} n_{\mathsf{hol}} \rceil$*. At the end of the interaction, either the verifier rejects or it outputs a coordinate* $z \in \mathbb{F}^m$*, and a value* $v \in \mathbb{F}$*, such that:*

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

We also define holographic *argument* systems, where soundness is relaxed to hold against bounded cheating provers:

Definition 3.23 (Holographic Interactive Proof (HIA))**.** *A holographic Interactive* Argument *for a language* $\mathcal{L}$ *is defined similarly to an* HIP *(Definition 3.22), except that all parties also get as input a security parameter* κ*, and* ε*-soundness is relaxed to hold only against cheating provers of size at most* $\mathsf{size}_{\mathcal{A}}(\kappa, n_{\mathsf{exp}}, n_{\mathsf{hol}})$*.*

HIA construction. We use a variant on the HIA of Amit and Rothblum [AR23] for bounded-depth circuits:

Theorem 3.24 (Holographic Argument [AR23])**.** *Assume* (λ, ε)*-secure one-way functions exist, and let* $\kappa = \kappa(n)$ *be a security parameter. Let* $\mathcal{L}$ *be a pair language with input length* $n_{\mathsf{exp}} = n_{\mathsf{exp}}(n) \leq n_{\mathsf{hol}} = n_{\mathsf{hol}}(n)$ *that is computable by log-space uniform circuits of fan-in 2, depth* $D = D(n)$ *and size* $S = S(n) \geq n_{\mathsf{hol}}, \kappa$*. For constructible field ensembles* $\mathbb{H}, \mathbb{F}$*, where* $|\mathbb{H}| \leq \mathsf{polylog}(S)$ *and* $\mathsf{polylog}(S) \leq |\mathbb{F}| \leq S$*, there is an* HIA *for* $\mathcal{L}$ *w.r.t* $\mathsf{LDE}_{\mathbb{H},\mathbb{F}}$ *. The protocol is public-coin with perfect completeness and has:*

- *round complexity:* $\mathsf{polylog}(S)$*,*
- *communication complexity:* $D \cdot \mathsf{poly}(\kappa, \log(S))$*,*
- *verifier runtime:* $D \cdot \mathsf{poly}(\kappa, \log(S)) + n_{\mathsf{exp}} \cdot \mathsf{polylog}(S)$*.*
- *honest prover runtime:* $\mathsf{poly}(S)$*.*
- $\left(\frac{\mathsf{polylog}(S)}{|\mathbb{F}|} + \varepsilon \cdot \mathsf{poly}(S)\right)$*-soundness against cheating provers of size* $\lambda(\kappa)/\mathsf{poly}(S)$*.*

The main theorem of [AR23] was stated for a *constant-round* protocol (a more challenging parameter regime), but allowed for n^{σ} communication and verifier runtime for an arbitrarily small constant σ. For a different choice of parameters, the same construction (or rather a simplification thereof) uses a polylogarithmic number of rounds, and the overhead is reduced to poly-logarithmic. We give an overview in Appendix A.

HIP construction. For unconditional results, we use the HIP of GKR:

Theorem 3.25 (Holographic IP [GKR15])**.** *Let* $\mathcal{L}$ *be a pair language with input length* $n_{\mathsf{exp}} = n_{\mathsf{exp}}(n) \leq n_{\mathsf{hol}} = n_{\mathsf{hol}}(n)$ *that is computable by log-space uniform circuits of fan-in 2, depth* $D = D(n)$ *and size* $S = S(n) \geq n_{\mathsf{hol}}$*. For constructible field ensembles* $\mathbb{H}, \mathbb{F}$*, where* $|\mathbb{H}| \leq \mathsf{polylog}(S)$ *and* $\mathsf{polylog}(S) \leq |\mathbb{F}| \leq S$*, there is an* HIP *for* $\mathcal{L}$ *w.r.t* $\mathsf{LDE}_{\mathbb{H},\mathbb{F}}$ *. The protocol is public-coin with perfect completeness and:*

- *round complexity:* $D \cdot \mathsf{polylog}(S)$,
- *communication complexity:* $D \cdot \mathsf{polylog}(S)$,
- *verifier runtime:* $(D + n_{\mathsf{hol}}) \cdot \mathsf{polylog}(S)$.
- *honest prover runtime:* $\mathsf{poly}(S)$.
- *soundness error:* $D \cdot \mathsf{polylog}(S)/|\mathbb{F}|$.

3.6 Interactive Proofs of Proximity

IPPs [EKR04, RVW13] are interactive proofs in which the verifier runs in sub-linear time in the input length, where the soundness requirement is relaxed to rejecting inputs that are *far* from the language w.h.p. (for inputs that are not in the language, but are close to it, no requirement is made). Interactive *arguments* of proximity relax the soundness requirement to hold against cheating provers of bounded size. We think of the input of the verifier as being composed of two parts: an *explicit* input $x \in \{0,1\}^{n_{\mathsf{exp}}}$ to which the verifier has direct access, and an *implicit* (longer) input $y \in \{0,1\}^{n_{\mathsf{imp}}}$ to which the verifier has oracle access. The goal is for the verifier to run in time that is sub-linear in n_{imp} and to verify that y is far from any y' such that the pair (x, y') are in the language. Since such languages are composed of input pairs, we refer to them as *pair languages*.

We define IPPs and IAPs below. In this work we focus on protocols where the verifier doesn't need to query the input during its interaction with the prover. After the interaction, the verifier rejects or it outputs a claim about a sublinear subset of the locations in the input. We call these *output-predicate* IPPs (and IAPs). There are many examples of such protocols, including any public-coin IPP (where the verifier's are just random coins) or protocols in the pre-coordinated model [GRS23], so long as the verifier's queries to the implicit input are non-adaptive.

Definition 3.26 (Interactive Proof of Proximity (IPP))**.** *An* interactive proof of proximity *(*IPP*) for the pair language $\mathcal{L}$ 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 $x \in \{0,1\}^{n_{\mathsf{exp}}}$, a soundness error $\varepsilon > 0$ and a proximity parameter $\delta > 0$. The verifier also gets oracle access to $y \in \{0,1\}^m$, whereas the prover has full access to y. The proof system gives the following guarantees:*

1. **Completeness:** *For every pair $(x, y) \in \mathcal{L}$, and parameters $\varepsilon, \delta > 0$ it holds that*

$$\Pr\left[\left(\mathcal{P}(y), \mathcal{V}^y\right)(x, |y|, \delta, \varepsilon) = 1\right] = 1.$$

2. **Soundness:** *For every $\varepsilon, \delta > 0$, $x \in \{0,1\}^n$ and y that is ε-far from the set $\{y' : (x, y') \in \mathcal{L}\}$, and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that*

$$\Pr\left[\left(\mathcal{P}^*(y), \mathcal{V}^y\right)(x, |y|, \delta, \varepsilon) = 1\right] \leq 1/2.$$

The verifier's runtime should be polynomial in n_{exp} and $1/\delta$ and sublinear in n_{imp} and $1/\varepsilon$.[5] *The query complexity $q = q(n_{\mathsf{exp}}, n_{\mathsf{imp}}, \delta, \varepsilon)$ bounds the number of queries the verifier makes to y. The communication complexity $\mathsf{cc} = \mathsf{cc}(n_{\mathsf{exp}}, n_{\mathsf{imp}}, \delta, \varepsilon)$ bounds the number of bits communicated between*

[5] The soundness error can be reduced by repetition, so the runtime is usually logarithmic in $1/\varepsilon$.

$\mathcal{V}$ *and* $\mathcal{P}$*. If the honest prover's running time is polynomial in* n_{exp} *and* n_{imp}*, then we way that the* IPP *is* doubly-efficient*.*

In an ***output-predicate*** IPP*, the verifier has the following special form: the verifier does not access the input during its interaction with the prover. At the end of the protocol, the verifier either rejects or it outputs a succinct subset* $Q \subseteq [n_{\mathsf{imp}}]$ *and a succinct predicate* ψ *(see Definition 3.9 and Definition 3.10). The verifier's decision (accept or reject) corresponds to whether* $\psi(x, y|_Q) = 1$ *(for acceptance). W.l.o.g. we usually let the verifier output the all-0 predicate rather than explicitly reject during the protocol. Here the query complexity is* $|Q|$*.*

Definition 3.27 (Interactive Argument of Proximity (IAP))**.** *An interactive* argument *of proximity is defined similarly to an* IPP*, but soundness only holds against bounded cheating provers.*

Formally, the definition is as in Definition 3.26, with the following modifications. The the two parties also get as input a security parameter κ*. The soundness property is relaxed, and only holds against cheating provers of size at most* $\mathsf{size}_{\mathcal{A}}(\kappa, n_{\mathsf{exp}}, n_{\mathsf{imp}})$*. The running time of the honest parties is polynomial in* κ*. Output-predicate* IAP*s are defined analogously.*

In this work we sometimes use IPPs and IAPs with an *absolute* proximity parameter (measuring the absolute distance of y from membership in the pair language w.r.t x) and *relative* query complexity (measuring the fractional density of the set Q within the implicit input).

IPP **Construction.** Our information-theoretic results use the IPP of [RVW13, RR20]. We use absolute distance and relative query complexity for compatibility with the honing protocols that we construct using IPPs:

Theorem 3.28 (IPP for Bounded Depth)**.** *Let* $\mathcal{L}$ *be a pair language decidable by a* $\rho = \rho(n_{\mathsf{imp}})$*-succinct circuit family of depth* $D = D(n_{\mathsf{imp}}) \geq \log(n_{\mathsf{imp}})$ *and size* $S = S(n_{\mathsf{imp}}) \geq n_{\mathsf{imp}}$ *(*$n_{\mathsf{imp}} \geq n_{\mathsf{exp}}$ *are the implicit and explicit input lengths).*

There is a public-coin output-predicate IPP *for* $\mathcal{L}$*: for absolute hamming distance* $\delta_{ABS} = \delta_{ABS}(n_{\mathsf{imp}}) \leq n_{\mathsf{imp}}$ *and desired soundness error* $\varepsilon' \geq 1/S$*, the protocol has:*

- Relative *query complexity:* $q = O((1/\delta_{ABS}) \cdot \log(1/\varepsilon'))$*.*
- *Communication Complexity:* $cc = (\delta_{ABS} + D) \cdot \mathsf{polylog}(S)$*.*
- *Round Complexity:* $D \cdot \mathsf{polylog}(S)$*.*
- *Verifier Running Time:* $(n_{\mathsf{exp}} + \rho + \delta_{ABS} + D) \cdot \mathsf{polylog}(S)$*.*
- *Prover Running Time:* $\mathsf{poly}(S)$*.*

At the end of the interaction either the verifier rejects or it outputs (as in Definition 3.26):

- *A* $\mathsf{polylog}(S)$*-succinct description* $\langle Q \rangle$ *of a set* $Q \subseteq [n_{\mathsf{imp}}]$ *of "surviving" indices of density* q*. The circuit computing this set has size* $n_{\mathsf{imp}} \cdot \mathsf{polylog}(S)$ *and depth* $\mathsf{polylog}(S)$*.*
- *a* $\mathsf{polylog}(n_{\mathsf{imp}})$*-succinct predicate* $\psi : \{0,1\}^{|Q|} \to \{0,1\}$ *computable by a circuit of size* $|Q| \cdot \mathsf{polylog}(S)$ *and depth* $\mathsf{polylog}(S)$*.*

Morover, the surviving indices in the set Q *are* 1*-wise uniform (this holds for any prover strategy, as the query set depends only on the verifier's coins).*

3.6.1 Extractable IPP

An *extractable interactive proof (or argument) of proximity* is an IPP (or IAP) where any cheating prover that convinces the verifier must "know" a close input in the language. "Knowing" means that the input can be efficiently extracted from the cheating prover: there is an *efficient* extractor Ext, such that for any cheating prover $\mathcal{P}^*$ that convinces the verifier with probability greater than the soundness error ε, the extractor (using black-box access to $\mathcal{P}^*$) produces y' s.t. $(x, y') \in \mathcal{L}$ and y' is δ-close to y.

We view extractable soundness as a natural notion that may be of independent interest. Several remarks are in order: first, extractable soundness implies the normal soundness notion of IPPs (or IAPs): if the input is far from the language, then no close input in the language can be extracted, so the proof system must be sound. In the other direction, any IPP or IAP is extractable using brute-force search: if the cheating prover breaks the IPP's soundness, this means that the input is not far from the language. We can find a close input in the language using brute-force search, but this is not an *efficient* (polynomial-time) extractor. Finally, we remark that we use extractable IPPs and IAPs as a building block in constructing computationally sound protocols: the extractor is used towards showing that a cheating prover breaks some underlying cryptographic assumption, and this is why it is important that it is efficient (regardless of whether the original soundness guarantee was information theoretic or computational).

Definition 3.29 (Extractable IPPs and IAPs)**.** *An extractable* IPP *for the pair language $\mathcal{L}$ is an interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ as in Definition 3.26, with the following modified soundness property:*

- **Extractable Soundness:** *There exists a polynomial-time extractor* Ext *such that for every $\varepsilon, \delta > 0$, $x \in \{0,1\}^{n_{\mathsf{exp}}}$, $y \in \{0,1\}^{n_{\mathsf{imp}}}$, and every (cheating) prover $\mathcal{P}^*$, if*

$$\Pr\left[\left(\mathcal{P}^*(y), \mathcal{V}^y\right)(x, n_{\mathsf{imp}}, \delta, \varepsilon) = 1\right] > \varepsilon,$$

  then $\mathsf{Ext}^{\mathcal{P}^*(\cdot)}(x, 1^{n_{\mathsf{imp}}}, 1^{1/\delta}, \varepsilon)$ outputs some y' that is δ-close to y such that $(x, y') \in \mathcal{L}$ with probability at least $1/2$, where $\mathsf{Ext}^{\mathcal{P}^*(\cdot)}$ denotes that Ext *has black-box access to $\mathcal{P}^*$. The success probability of the extractor is over its own coins and can be amplified by repetition.*

Similarly, an output-predicate *extractable* IPP *is an output-predicate* IPP *(as in Definition 3.26), with the above modified soundness guarantee. In particular, if the probability that the verifier outputs a predicate that accepts (the sublinear subset of) the input is larger than ε, then the extractor should recover a close input in the language.*

An extractable IAP *(or output-predicate extractable* IAP*) is defined as in Definition 3.27, but with the above extractable soundness guarantee. I.e., the extractor is only guaranteed to work for* computationally bounded *cheating provers that can make the verifier accept w.p. greater than ε.*

First-message extractor. A particularly strong form of extractability is when the extractor does not need oracle access to the cheating prover, but can extract a valid input y' solely from observing the first message sent by the prover. Following the prover's first message, either its probability of convincing the verifier is at most ε, or the extractor can extract a close input in the language. This strong extractability notion simplifies the security analysis of (computationally sound) protocols that use the IPP or the IAP as a building block.

Definition 3.30 (First-Message Extractable IPP)**.** *A* first-message extractable IPP *for the pair language $\mathcal{L}$ is an interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ as in Definition 3.26, with the following modified soundness property:*

- **First-Message Extractable Soundness:** *There exists a polynomial-time extractor* Ext *such that for every $\varepsilon, \delta > 0$, $x \in \{0,1\}^{n_{\mathsf{exp}}}$, $y \in \{0,1\}^{n_{\mathsf{imp}}}$, and every (cheating) prover $\mathcal{P}^*$, if the first message $\alpha = \mathcal{P}^*(x, n_{\mathsf{imp}}, 1^{1/\delta}, \varepsilon)$ leads to a high probability that $\mathcal{V}$ accepts, i.e. if:*

$$\Pr\left[\left(\mathcal{P}^*(y), \mathcal{V}^y\right)(x, n_{\mathsf{imp}}, \delta, \varepsilon) = 1 \mid \textit{first message} = \alpha\right] > \varepsilon,$$

  then $\mathsf{Ext}(\alpha, x, n_{\mathsf{imp}}, 1^{1/\delta}, \varepsilon)$ *outputs some y' that is δ-close to y such that $(x, y') \in \mathcal{L}$ with probability at least $1/2$. Note that* Ext *takes only the first message α as input and does not require black-box access to $\mathcal{P}^*$.*

As before, an output-predicate *first-message extractable* IPP *is an output-predicate* IPP *(as in Definition 3.26), with the above modified soundness guarantee.*

A first-message extractable IAP *(or output-predicate first-message extractable* IAP*) is defined as in Definition 3.27, but with the above extractable soundness guarantee. Here we do require the extractability guarantee to hold for* any *first message, and for any bounded prover strategy in the subsequent rounds.*[6] *I.e. we require extractable soundness to hold for any first message, and every bounded strategy that the prover can employ in subsequent rounds.*

Constructions. The IPPs of [RVW13, RR20] (which natively produce an output predicate) can be modified to achieve first-message extractability. The main idea is for the prover to send a $(\delta \cdot n_{\mathsf{imp}})$-checksum for y in its first message. The prover an the verifier run an IPP that checks both membership in $\mathcal{L}$ and that y is consistent with the checksum. There is at most one input y' within a δ-radius of y satisfying the checksum. The extractor efficiently decodes this y' from the prover's first message. Soundness follows because if $(x, y') \notin \mathcal{L}$, then y is far from the language that the IPP checks w.r.t the explicit input x (note that this is the case even if y is close to some other y'' s.t. $(x, y'') \in \mathcal{L}$), and the IPP verifier will reject w.h.p. A similar construction works for ensuring first-message extractability for IAPs, where we note that indeed the extractability guarantee holds for *any* first-message (namely any checksum), so long as the subsequent argument system is sound against cheating provers running within the specified computational bounds. This transformation incurs a $\widetilde{O}(\delta \cdot n_{\mathsf{imp}})$ communicaiton overhead for sending the checksum, but this is smaller than the communication complexity of the protocols we use.

See Section 3.6.3 for formal statements of the results we can get.

3.6.2 Nested IPPs

In a *nested* IPP, there are m explicit-implicit input pairs $(x_i \in \{0,1\}^{n_{\mathsf{exp}}}, y_i \in \{0,1\}^{n_{\mathsf{imp}}})_{i \in [m]}$, and a sequence of m claims about nested subsets (see Definition 3.1) $A_m \subseteq \ldots \subseteq A_1 \subseteq [n_{\mathsf{imp}}]$ of locations in the corresponding implicit inputs. One could run m independent output-predicate IPPs to verify these claims, where the i-th IPP verifies the claim about $(x_i, y_i|_{A_i})$, returning a set $Q_i \subseteq A_i$ and a predicate ψ_i. However, this would mean that the output subsets (Q_i) are unrelated, and might

[6] We focus on this definition because it is achieved by natural constructions. However, one could also relax the requirement to focus on bounded provers that produce the first message.

not preserve the nested structure. A nested IPP guarantees that the output claims are also about nested subsets. We restrict our attention to the case where all the claims are with respect to the same *absolute* proximity parameter, so the relative query complexity is the same for each of the claims. We focus on output-predicate IPPs and IAPs throughout, as we are concerned with the structure of the output subsets. For convenience, we also focus on first-message extractability (the definition can be extended to other soundness notions).

In the definition below, we consider "universal" protocols that get a description of the relevant pair language as part of their input (this analogous to a universal Turing Machine or circuit, which gets a description of the machine or circuit, respectively). The pair language is described via a succinct predicate that takes the explicit input and (a subset of) the implicit input and accepts or rejects. The protocol should work for any predicates within a pre-specified family Φ.

Definition 3.31 (Nested IPP or IAP)**.** *A* nested (first-message extractable) IPP *for a family of predicates Φ is an interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ with the following inputs:*

- *m explicit inputs $x_1, \ldots, x_m \in \{0,1\}^{n_{\mathsf{exp}}}$,*
- *m implicit inputs $y_1, \ldots, y_m \in \{0,1\}^{n_{\mathsf{imp}}}$ (only the prover has access to the y_i's),*
- *an absolute proximity parameter $\delta_{\mathsf{ABS}} \in \mathbb{N}$ and soundness error $\varepsilon > 0$,*
- *m nested succinct input sets $A_m \subseteq \ldots \subseteq A_1 \subseteq [n_{\mathsf{imp}}]$,*
- *m succinct input predicates $\phi_1, \ldots, \phi_m \in \Phi$.*

At the end of the protocol, the verifier outputs a succinct subset $Q \subseteq [n_{\mathsf{imp}}]$. For each $i \in [m]$ let $Q_i = A_i \cap Q$. The m (succinct) output sets are $Q_m \subseteq \ldots \subseteq Q_1 \subseteq [n_{\mathsf{imp}}]$. The verifier also outputs m succinct output predicates $\psi_1, \ldots, \psi_m$. We require:

- **Completeness:** *For every $i \in [m]$, if $\phi_i(x_i, y_i|_{A_i}) = 1$, then*
$$\Pr\left[\psi_i(x_i, y_i|_{Q_i}) = 1\right] = 1.$$
- **First-Message Extractable Soundness:** *There exists a polynomial-time extractor* Ext *such that for every $\delta_{\mathsf{ABS}} \in \mathbb{N}$, $\varepsilon > 0$, inputs $(x_i, y_i)_{i\in[m]}$, $i \in [m]$, and every (cheating) prover $\mathcal{P}^*$, if the prover's first message α leads to a high probability that ψ_i is true:*
$$\Pr\left[\psi_i(x_i, y_i|_{Q_i}) = 1 \mid \textit{first message} = \alpha\right] > \varepsilon,$$
then with probability at least $1/2$, $\mathsf{Ext}(\alpha, x_i, n_{\mathsf{imp}}, i, 1^{\delta_{\mathsf{ABS}}}, \varepsilon)$ outputs some y' that is at absolute distance at most δ_{ABS} from y_i such that $\phi_i(x, y'|_{A_i}) = 1$.

A nested (first-message extractable) IAP *is defined analogously: the parties also get a security parameter κ in their inputs (see Definition 3.27), and soundness is relaxed to only hold for cheating provers of a specified size bound $\mathsf{size}_{\mathcal{A}}(\kappa, n_{\mathsf{exp}}, n_{\mathsf{imp}})$. As in first-message extractable* IAP*s (Definition 3.30), the extractability guarantee to holds for* every *first message sent by the cheating prover, so long as a bounded prover strategy convinces the verifier to accept w.p. ε in the subsequent rounds.*

The (relative) query complexity of a nested IPP is (the worst-case bound on) $|Q|/n_{\mathsf{imp}}$.The choice of indices in Q is always at least 1-wise uniform, so for each $i \in [m]$ the expected fractional density of Q_i in A_i is (at most) $|Q|/n_{\mathsf{imp}}$. We remark that it is natural to run nested IPPs when the implicit inputs are all equal, but we also allow for running nested IPPs for differing implicit inputs.

Constructions. Any IPP or IAP where the query set is just a function of the verifier's coin tosses (and the absolute distance parameter) can be made into a nested IPP as follows. Run in parallel m separate IPPs, one for each of the claims, where all executions use the same coin tosses for the verifier. In the i-th execution, the implicit input is y'_i, where all of the bits of y_i that are outside the set A_i are set to 0 (and the bits in A_i are as in y_i), and the protocol checks that $\phi_i(x_i, y'_i)$ accepts (we abuse notation and allow ϕ_i to take the entire input y'_i, where it ignores the bits outside A_i). Note that the absolute distance of y'_i from satisfying $\phi_i(x_i, \cdot)$ is the same as the absolute distance of y_i. The IPPs all use the same coins, so the query set Q is the same in all of them. The i-th IPP outputs a predicate ψ_i over $y'_i|_Q$. The nested verifier outputs ψ'_i which sets the bits of the input that are outside A_i to 0 and then checks if ψ_i accepts. Completeness and extractable soundness now follow by construction (the extractor uses the first message sent for the i-th protocol to extract a close satisfying input input for that predicate). By construction, the query set for the i-th output predicate ψ_i is exactly $Q \cap A_i$ (since all bits outside A_i are set to 0). See Section 3.6.3 for a formal statement.

3.6.3 Extractable Nested IPPs: Protocols

As described above, the IPPs of [RVW13, RR20] can be transformed into nested and first-message extractable IPPs. To reduce the round complexity, we use an IAP derived from the protocol of [AR23], see Theorem 3.24. This gives the following result for uniform bounded-depth circuits:

Theorem 3.32 (IAP from [AR23])**.** *Assume there exist (λ, ε)-secure one-way functions. Let Φ be the class of $\rho = \rho(n_{\mathsf{imp}})$-succinct predicates of depth $D = D(n_{\mathsf{imp}}) \geq \log(n_{\mathsf{imp}})$ and size $S = S(n_{\mathsf{imp}}) \geq n_{\mathsf{imp}}$ (where $n_{\mathsf{imp}} \geq n_{\mathsf{exp}}$ are the implicit and explicit input lengths).*

There is a public-coin first-message extractable nested IAP *for Φ. For absolute hamming distance $\delta_{ABS} = \delta_{ABS}(n_{\mathsf{imp}}) \leq n_{\mathsf{imp}}$, security parameter $\kappa = \kappa(n_{\mathsf{imp}}) \leq S$, desired soundness error $\varepsilon' \geq 1/S$, and $m = m(n_{\mathsf{imp}}) \in \mathbb{N}$ nested inputs, where the sets A_i are ρ-succinct, the protocol has:*

- *Expected relative query complexity:* $q = O((1/\delta_{ABS}) \cdot \log(1/\varepsilon'))$.
- *Communication Complexity:* $cc = m \cdot (\delta_{ABS} + D) \cdot \mathsf{poly}(\kappa, \log(S))$.
- *Round Complexity:* $\mathsf{polylog}(S)$.
- *Verifier Running Time:* $m \cdot (n_{\mathsf{exp}} + \rho + \delta_{ABS} + D) \cdot \mathsf{poly}(\kappa, \log(S))$.
- *Prover Running Time:* $m \cdot \mathsf{poly}(S)$.
- *Soundness: the* IAP *is $(\varepsilon' + \varepsilon \cdot \mathsf{poly}(S))$-sound against adversaries of size $\lambda(\kappa)/\mathsf{poly}(m, S)$.*

At the end of the interaction either the verifier rejects or it outputs (as in Definition 3.31):

- *A $\mathsf{polylog}(S)$-succinct description $\langle Q \rangle$ of a set $Q \subseteq [n_{\mathsf{imp}}]$ of "surviving" indices of expected density q. The circuit computing this set has size $n_{\mathsf{imp}} \cdot \mathsf{polylog}(S)$ and depth $\mathsf{polylog}(S)$.,*
- *m descriptions $(\langle \psi_i \rangle)_{i \in [m]}$ of $\mathsf{polylog}(S)$-succinct predicates $\left(\psi_i : \{0,1\}^{|Q \cap A_i|} \to \{0,1\}\right)_{i \in [m]}$. The circuit computing ψ_i has size $|Q \cap A_i| \cdot \mathsf{polylog}(S)$ and depth $\mathsf{polylog}(S)$.*

Morover, the surviving indices in the set Q are 1-wise uniform (this holds for any prover strategy, as the query set depends only on the verifier's coins).

Notes about the construction. The IAP is derived from the HIA of Theorem 3.24. We can derive a regular IAP as follows:

- Following the template from [RVW13], we first use the holographic argument to reduce checking proximity to $\mathcal{L}$ to checking proximity to a PVAL instance.
- We then run the (unconditionally sound) IPP for PVAL from [RR20].

Moving from a regular IAP to a nested first-message extractable IAP is done as described in Section 3.6.1 and 3.6.2 (the "constructions" paragraph at the end of each section). □

3.6.4 Running IPPs on configuration tables

In this work we run nested IPP on input tables that include configurations of the machine $\mathcal{M}$ from the computation we are trying to prove and verify. When this is the case, the input is a "tall" and "skinny" matrix, where each row corresponds to a certain configuration (of size $O(S)$), with potential auxiliary information. We find it helpful to work directly with absolute *row* distance on such tables:

Definition 3.33 (Absolute row distance)**.** *The absolute row distance between a table U and a predicate ϕ is:*

$$\Delta_{\mathsf{ABS}}(U,\phi) = \min_{U':\,|U|=|U'|,\,\phi(U')=1} \{\#\text{ rows on which } U,\,U' \text{ differ}\}. \tag{1}$$

Remark 3.34 (Row distance and Hamming distance)**.** *When we run the* IPP *of Theorem 3.32 on a table of configurations to check that the table satisfies a predicate ϕ, we implicitly first encode each configuration, and run an* IPP *checking a new predicate ϕ', which checks that all configurations in the input are properly encoded and then that their decoding satisfies ϕ. The encoding ensures that the encoded input's* Hamming *distance from satisfying ϕ' is $\Theta(|\mathsf{row}| \cdot \delta_{\mathsf{ABS}}(U,\phi))$. We then run the* IPP *with this absolute Hamming distance, which only incurs communication and verifier time blowups that are linear in the width of a row.*

3.7 Honing Protocols

A honing protocol is a protocol between a prover and a verifier, where both take as input an instance for the multi-tableau language, $(\mathcal{M}, x, S, \ell, (A_i)_{i=1}^m, (\phi_i)_{i=1}^m)$ (the verifier gets implicit access to the collection of configurations). The goal is refining the search for inconsistencies within the computation tableaux to smaller sets: the verifier outputs a sequence of (succinct) nested subsets of the configurations in the A_i's and a sequence of corresponding (succinct) predicates. If for some $i \in [m]$, the configurations in A_i are far from any table whose induced tableau satisfy ϕ_i, then the protocol's output should induce a significantly smaller set and a corresponding predicate that rejects (the concatenation of) that set's tableaux. Intuitively, this is "progress" because the inconsistency is on a shorter overall computation (and so we can hope it is easier to detect). Distance is measured using the absolute row distance (Definition 3.33) between U and the predicate that expands the tableaux and then checks whether ϕ accepts. I.e. the number of rows we'd need to change in U to get a table U' whose induced tableau satisfy ϕ.

We parameterize a honing protocol by the computation length of the tableaux it can handle, and by a "honing parameter" η that gives a bound on how much smaller the output set containing

the inconsistency will be. We emphasize that we are interested in protocols where the verifier's runtime is much smaller than $(\sum_i |A_i|)$ (and much smaller than the computation length ℓ). In particular, the verifier treats the A_i's as an implicit input and outputs succinct descriptions of the subsets it chooses.

Looking ahead, a honing protocol is a type of nested IPP (with minor differences), but we find it helpful to provide an explicit definition. We focus on computational soundness against non-uniform adversaries. The definition can be adapted to unconditional soundness. See Remarks 3.36 and 3.37.

Definition 3.35 ($(\eta, \delta_{\mathsf{ABS}}, \ell)$-Honing Protocol)**.** *A* honing protocol *is a protocol between an untrusted prover and a (sublinear) verifier. It has the following parameters, which can all be functions of the input length: honing parameter $\eta \in (0,1]$, absolute proximity parameter $\delta_{\mathsf{ABS}} \in \mathbb{N}$, computation length $\ell \in \mathbb{N}$, completeness and soundness errors $\varepsilon_c, \varepsilon_s : \mathbb{N} \to [0,1]$ and a bound $\mathrm{s}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ on the adversary's size.*[7] *The input includes a security parameter κ and an instance for the multi-tableau language $(\mathcal{M}, x, 1^S, \ell, U, (A_i)_{i=1}^m, (\phi_i)_{i=1}^m)$. The verifier gets implicit access to the table U of configurations and explicit access to $\mathcal{M}, x, 1^S, \ell, 1^\kappa$, to the succinct descriptions of the sets (A_i) and the predicates (ϕ_i), and to the security parameter.*

The verifier does not query the table U during the protocol. At the end of its interaction with the prover, it either rejects, or it outputs a succinct a subset $B \subseteq [|U|]$ of "surviving rows" and predicates $(\psi_i)_{i=1}^m$. The protocol guarantees:

- ***η-honing:*** *For every $j \in [|U|]$:*

$$\Pr[j \in B] \leq \eta, \tag{2}$$

  where the probability is only over the coin tosses of the verifier (so Equation (2) holds for every prover strategy, cheating or honest).

  For every $i \in [m]$, $B_i = A_i \cap B$: i.e. surviving rows survive in all subsets (and non-surviving ones are dropped from every subset). This structure implies that if the A_i's are nested sets, then so are the B_i's. Moreover, the expected size of each B_i is at most $\eta \cdot |A_i|$.

- ***Completeness:*** *If for all $i \in [m]$, $\phi_i\left(\mathcal{T}_{U|_{A_i},\ell}\right) = 1$, then with probability at least $1 - \varepsilon_c(\kappa)$, for all $i \in [m]$, $\psi_i\left(\mathcal{T}_{U|_{B_i},\ell}\right) = 1$.*

- ***Proximity-soundness:*** *For every non-uniform cheating adversary $\mathcal{A}$ of size at most $\mathrm{s}_{\mathcal{A}}(\kappa)$, for every input and every index $i \in [m]$, if $U|_{A_i}$ is at absolute row distance at least δ_{ABS} from a table that induces tableaux satisfying ϕ_i, then the probability that for every $j \in [i,m]$ it holds that $\psi_j\left(\mathcal{T}_{U|_{B_j},\ell}\right) = 0$ is at most $\varepsilon_s(\kappa)$.*

The above probabilities are over the verifier's and the prover's (or the adversary's) coin tosses.

Several remarks are in order:

Remark 3.36 (Variations on the definition)**.** *The above definition assumes the subsets A_i are nested (see Definition 3.1), but the definition can be adapted to general subsets (in the general case, the subsets B_i will also not necessarily be nested). The definition can also be generalized to have*

[7] We focus on soundness against non-uniform adversaries. For uniform soundness, $\mathrm{s}_{\mathcal{A}}$ bounds their runtime.

unrelated sets of surviving rows in the different A_i's (rather that having a "global" decision on whether a row survives, i.e. is in the set B, which determines its survival in all the A_i's). Another natural variation is soundness against unbounded adversaries: in this case we call the protocol a proof *system, or refer to it as* unconditionally sound*: we construct such protocols in Section 5.*

Remark 3.37 (Honing protocols and nested IPP.)**.** *Any honing protocol is (almost) an output-predicate nested* IAP *(Definition 3.31). Syntactically, there is a minor difference in the soundness requirement of a honing protocol: if the i-th input is far from the predicate, then we allow for the false output claim to be on any set $j \in [i, m]$, whereas requiring it to be on the i-th output claim in the general definition of a nested* IPP*, but we don't view this as a meaningful distinction (indeed, it is possible to incorporate all the claims $(\psi_j)_{j\in[i,m]}$ into a single claim about $U|_{B_i}$). There is a contextual difference in that the output predicates of* IPP*s are usually quite efficient to compute, whereas for a honing protocol the output claim still requires computing tableaux of length ℓ, which can be quite large.*

We view honing protocols as a natural object, and they play a pivotal role in our constructions, so we view them as a type of IPP *that is worth defining and studying in its own right. We note that we use nested* IPP*s in our recursive construction of honing protocols: both in the base of the recursion, and in the recursive step (see Sections 2 and 4).*

4 Main Construction

We detail our recursive honing protocol construction, and its application to constructing arguments.

Organization The base of the recursion is in Section 4.1. The recursive step and its analysis are in Section 4.2. We put things together to obtain the resulting honing protocol in Section 4.3, and we derive an efficient argument system in Section 4.4.

4.1 Base of The Recursion

Corollary 4.1. *Assume there exists a one-way function secure against $\lambda(\kappa)$-size adversaries. There is a honing protocol for the multi-tableau language as follows: for (absolute) distance $\delta^{\mathsf{base}}_{ABS}$, an instance $(\mathcal{M}, x, 1^S, \ell^{\mathsf{base}}, U, (A_i)_{i\in[m]}, (\phi_i)_{i\in[m]})$, take $T = |U| \cdot \ell^{\mathsf{base}}$. We require that the sets A_i and predicates ϕ_i are ρ^{base}-succinct with depth $\mathsf{poly}(\kappa, \log T)$ and size $\mathsf{poly}(T)$, that $|x|, S, m, \kappa, \rho^{\mathsf{base}} \leq T$. The protocol is public-coin with perfect completeness. For desired soundness $\varepsilon \geq \mathsf{poly}(1/\kappa)$, it is ε-sound against adversaries of size $(\lambda(\kappa)/\mathsf{poly}(T))$ size and has:*

- ***Honing parameter:*** $\eta^{\mathsf{base}} = O\left(\frac{\log(1/\varepsilon)}{\delta^{\mathsf{base}}_{ABS}}\right)$.
- ***Communication complexity:*** $\mathsf{cc}^{\mathsf{base}} = m \cdot \delta^{\mathsf{base}}_{ABS} \cdot S \cdot \ell^{\mathsf{base}} \cdot \mathsf{poly}(\kappa, \log(T))$.
- ***Round complexity:*** $r^{\mathsf{base}} = \mathsf{polylog}(T)$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time}^{\mathsf{base}} = m \cdot \delta^{\mathsf{base}}_{ABS} \cdot (|x| + \rho^{\mathsf{base}}) \cdot \mathsf{poly}(\ell^{\mathsf{base}}, \kappa, \log(T))$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time}^{\mathsf{base}} = \mathsf{poly}(T)$.

The output sets B_i and predicates ψ_i are $\mathsf{polylog}(T)$-succinct, and computable by circuits of depth $\mathsf{polylog}(T)$ and size $|U| \cdot S \cdot \mathsf{polylog}(T)$ (for B_i) and $|B_i| \cdot S \cdot \mathsf{polylog}(T)$ (for ψ_i). We emphasize that he succinctness and complexities of these circuits are independent of those of the input predicate ϕ.

Proof. The corollary follows from Theorem 3.32 by viewing the honing protocol as a nested IAP (see Remark 3.37). For each $i \in [m]$, the i-th predicate ϕ'_i for the nested IAP first expands each state in U to the ℓ^{base}-length tableau starting at that state, and then applies the predicate ϕ_i for the honing protocol to the resulting tableaux. The tableau expansion is performed by the standard $O(\log(T))$-space-uniform circuit of depth $O(\ell^{\mathsf{base}} \cdot \log T)$ and size $O(\ell^{\mathsf{base}} \cdot S \cdot \log T)$ that takes a state and outputs the ℓ^{base} states that follow it in the Turing machine's computation. By construction, the composed predicates ϕ'_i are ρ^{base}-succinct and have $(\ell^{\mathsf{base}} \cdot \mathsf{polylog}(T))$ depth and $\mathsf{poly}(T)$ size. The complexity bounds follow directly from those of Theorem 3.32. □

4.2 The Recursive Step

We formalize the recursive step of the honing protocol. The main theorem of this section shows how to construct a protocol for computations of length $(k \cdot \ell)$ using a single oracle call to a protocol for computations of length ℓ. The construction is in Figure 1. We prove its soundness and honing properties in Lemma 4.3 and analyze its complexity in Lemma 4.2.

We remark that an important parameter throughout this construction is the total size T of the computation being proven: the table U is of size $|U|$, and each of its entries induces a length-ℓ^{outer} tableau that is given as input to the predicates ϕ_i^{outer}, so we view the total computation size as $T = |U| \cdot \ell^{\mathsf{outer}}$ and use this quantity to control several other parameters.

Checksum-UOWHF subprotocol. In Step 1 of the protocol of Figure 1, the prover and the verifier engage in a "checksum-UOWHF" commitment protocol. For each subset i (ignoring the i subscript), the prover first sends a checksum that "binds" it (in some sense) to a large implicit table $\widetilde{U}$: the unique table U that agrees with the checksum and differs from U in at most d rows. After this step is performed, the verifier reveals the keys to the hash functions used for the tree commitment (Definition 3.7): for each row in U, the prover computes the LDE encoding of its tableau (see Section 3.4.1 for the definition of an encoded tableau), and then computes the hash root of the tree applied to that LDE. This gives a corresponding table of hash roots, and the prover also sends a checksum for that table. We spell this out here to emphasize the order of events: since the UOWHF is only targeted-collision resistant, it is important that the matrix $\widetilde{U}$ was well defined and efficiently computable *before* the verifier reveals its hash keys.

Lemma 4.2 (Recursive construction complexity)**.** *For every input tuple $(\mathcal{M}, x, 1^S, \ell^{\mathsf{outer}}, U)$ and ρ^{outer}-succinct sets $(A_i^{\mathsf{outer}})_{i\in[m]}$ and predicates $(\phi_i^{\mathsf{outer}})_{i\in[m]}$, for parameters κ, d and $k \le \ell^{\mathsf{outer}}$, let $T = |U| \cdot \ell^{\mathsf{outer}}$. We assume that $T \ge |x|, S, \kappa, d, m, \rho^{\mathsf{outer}}$. We also assume that the circuits A_i^{outer} and predicates ϕ^{outer_i} have $\mathsf{poly}(T)$ size and $\mathsf{poly}(\log(T), \kappa)$ depth.*

Let $\mathsf{cc}^{\mathsf{inner}}, r^{\mathsf{inner}}, \mathcal{V}\mathsf{time}^{\mathsf{inner}}, \mathcal{P}\mathsf{time}^{\mathsf{inner}}$ denote the communication, rounds and verifier and prover runtimes (respectively) of the inner honing protocol when it is run as follows:

- *the parameters are η^{inner}, $\delta_{ABS}^{\mathsf{inner}}$, $|x|$, S, ℓ^{outer}/k, $|U'^{\mathsf{expd}}| \le (k \cdot |U| \cdot (1 + 1/k))$, $(m + 1)$,*
- *the sets A_i^{inner} and the predicates ϕ_i^{inner} are $\rho^{\mathsf{inner}} = (\rho^{\mathsf{outer}} + m \cdot S \cdot \mathsf{poly}(\kappa, k, d, \log T))$-succinct.*

Outer protocol Π^{outer}, with a recursive call to inner protocol Π^{inner}

Parameters: honing η^{inner}, absolute proximity $\delta^{\mathsf{inner}}_{\mathsf{ABS}}$, security κ, expansion k, checksum d.
Input: $(\mathcal{M}, x, 1^S, \ell^{\mathsf{outer}} = (k \cdot \ell), U,$ nested sets $(A_i^{\mathsf{outer}})_{i=1}^m, (\phi_i^{\mathsf{outer}})_{i=1}^m)$. Take $T = |U| \cdot \ell^{\mathsf{outer}}$.

1. **Checksum-UOWHF commitments:** For each $i \in [m]$ (in parallel): the prover sends a d-checksum $\widetilde{C_{U\,i}}$ for the table $U|_{A_i^{\mathsf{outer}}}$, the verifier then chooses and sends a UOWHF-tree key h (the same key for all $i \in [m]$), the prover computes the matrix R of hash tree roots for the encodings of the tableaux of U, and sends a d-checksum $\widetilde{C_{R\,i}}$ for $R|_{A_i^{\mathsf{outer}}}$.

2. **Sampling subsets:** The verifier picks a subset A^{samp} of U's rows: each row is chosen w.p. $1/2k$ and the choices are $10k \log\log T$-wise indep. If $|A^{\mathsf{samp}}| \geq (1 + 1/k)|U|$, then the verifier accepts. Otherwise, $\forall i \in [m]$, $A_i^{\mathsf{samp}} = A_i^{\mathsf{outer}} \cap A^{\mathsf{samp}}$. The prover sends d-checksums: $\widetilde{C_{U\,i}}^{\mathsf{samp}}$ for $U|_{A_i^{\mathsf{samp}}}$ and $\widetilde{C_{R\,i}}^{\mathsf{samp}}$ for $R|_{A_i^{\mathsf{samp}}}$.

3. **Recursive Call:** Define $U' = \begin{pmatrix} U \\ U|_{A^{\mathsf{samp}}} \end{pmatrix}$. We index the rows of U' using the disjoint union $[|U|] \coprod A^{\mathsf{samp}}$ (see Definition 3.2). Let $(U')^{\mathsf{expd}}$ be the k-fold expansion of U' (see Definition 3.16). For each $i \in [m]$, let A_i^{expd} and $A_i^{\mathsf{samp-expd}}$ be the k-fold index expansions of A_i^{outer} and A_i^{samp} (respectively). For $i \in [m+1]$, define $A_i^{\mathsf{inner}} = A_i^{\mathsf{expd}} \bigcup A_{i-1}^{\mathsf{samp-expd}}$ (define $A_0^{\mathsf{samp-expd}}$ and A_{m+1}^{expd} to be empty), and let ϕ_i^{inner} be a predicate on the ℓ-length tableaux of $(U')^{\mathsf{expd}}|_{A_i^{\mathsf{inner}}}$ that checks:

   (a) ϕ_i^{outer} accepts the concatenation of the tableaux of the states in $(U')^{\mathsf{expd}}|_{A_i^{\mathsf{expd}}}$.

   (b) each k-row block of $(U')^{\mathsf{expd}}|_{A_i^{\mathsf{inner}}}$ corresponds to a single longer computation of length $(k \cdot \ell)$: the j-th state in the block is the $((j-1) \cdot \ell)$-th state of the longer computation.

   (c) The checksum $\widetilde{C_{U\,i-1}}^{\mathsf{samp}}$ is consistent with $(U')^{\mathsf{expd}}|_{A_{i-1}^{\mathsf{samp-expd}}}$ (on the first row of each k-block).

   (d) The checksum $\widetilde{C_{R\,i-1}}^{\mathsf{samp}}$ is consistent with hash roots computed on the encodings of the $(k \cdot \ell)$-length tableaux in $(U')^{\mathsf{expd}}|_{A_{i-1}^{\mathsf{samp-expd}}}$ (one root for each k-row block in $A_{i-1}^{\mathsf{samp-expd}}$).

   The prover and verifier run Π^{inner} on $(\mathcal{M}, x, 1^S, \ell, (U')^{\mathsf{expd}}, (A_i^{\mathsf{inner}})_{i=1}^{m+1}, (\phi_i^{\mathsf{inner}})_{i=1}^{m+1})$ with parameters $\eta^{\mathsf{inner}}, \delta^{\mathsf{inner}}_{\mathsf{ABS}}$. Let $B^{\mathsf{inner}} \subseteq [|U|]$ and $(\psi_i^{\mathsf{inner}})_{i=1}^{m+1}$ be the output set and LDE-predicates.

   For $i \in [m+1]$, let $B_i^{\mathsf{ctrt}} \subseteq A_i^{\mathsf{outer}}$ be the k-fold index contraction of $(B^{\mathsf{inner}} \cap A_i^{\mathsf{expd}})$, and let $B_i^{\mathsf{samp-ctrt}} \subseteq A_i^{\mathsf{samp}}$ be the k-fold index contraction of $(B^{\mathsf{inner}} \cap A_i^{\mathsf{samp-expd}})$ (see Definition 3.16). The prover splits the LDE-predicate ψ_i^{inner} into LDE-predicates ψ_i^{ctrt} on the length-$(k \cdot \ell)$ tableaux of $U|_{B_i^{\mathsf{ctrt}}}$ and $\psi_{i-1}^{\mathsf{samp-ctrt}}$ on the length-$(k \cdot \ell)$ tableaux of $U|_{B_{i-1}^{\mathsf{samp-ctrt}}}$ (see Proposition 3.20).

4. **Holographic argument:** For each $i \in [m]$, the prover and verifier run an HIA on the matrix of $(k \cdot \ell)$-length tableaux induced by $U|_{B_i^{\mathsf{ctrt}}}$, to check that ψ_i^{ctrt} accepts, with soundness error $\varepsilon^{\mathsf{hol}} = 1/\mathsf{polylog}(T)$. This gives an output claim $(z_i^{\mathsf{hol}}, v_i^{\mathsf{hol}})$ about a single coordinate in the LDE of these tableaux. Decomposing the LDE into one row per element of B_i^{ctrt}, the z_i^{hol}-th coordinate is a linear combination of the z_i-th coordinate in the LDE of each row (see Section 3.4.1).

5. **The consistency IAP:** Let M be a $|U|$-row matrix, where $M[j] = (U[j], R[j], (v_{i,j}, \mathsf{open}_{i,j})_{i \in [m]})$: $v_{i,j}$ is the value of the z_i-th coordinate in the LDE of the $(k \cdot \ell)$-length tableau starting at $U[j]$, and $\mathsf{open}_{i,j}$ is the opening of this value w.r.t. the root $R[j]$. The prover and verifier run a nested IAP with *absolute* proximity parameter d, *relative* query complexity $\widetilde{O}(1/d)$ and soundness $\varepsilon^{\mathsf{IAP}} = 1/\mathsf{polylog}(T)$. The i-th execution in the nested IAP is for the subset A_i^{outer} and checks that $M|_{A_i^{\mathsf{outer}}}$ is close to the language L_i^{IAP} (Definition 4.8): that all the openings are correct, that the linear combination of the $v_{i,j}$'s is v_i^{hol} and that $M|_{A_i^{\mathsf{outer}}}$ is consistent with $\widetilde{C_{U\,i}}, \widetilde{C_{R\,i}}, \widetilde{C_{U\,i}}^{\mathsf{samp}}, \widetilde{C_{R\,i}}^{\mathsf{samp}}$. Let $B_i^{\mathsf{IAP}} \subseteq A_i^{\mathsf{outer}}$ be the set of rows queried by the i-th IAP and let ψ_i^{IAP} be the IAP's output predicate. We view ψ_i^{IAP} as a predicate over the $(k \cdot \ell)$-length tableaux of rows in B_i^{IAP}, see Remark 4.13.

6. **Output:** The i-th output set is $B_i^{\mathsf{outer}} = B_i^{\mathsf{IAP}} \bigcup B_i^{\mathsf{samp-ctrt}}$. The i-th output LDE-predicate ψ_i^{outer} is obtained by running a final holographic argument on the $(k \cdot \ell)$-length tableaux of states in $U|_{B_i^{\mathsf{outer}}}$, checking that $\psi_i^{\mathsf{samp-ctrt}}$ accepts (tableaux of) $U|_{B_i^{\mathsf{samp-ctrt}}}$ and that ψ_i^{IAP} accepts (tableaux of) $U|_{B_i^{\mathsf{IAP}}}$.

Figure 1: Recursive Outer Protocol Π^{outer}

Moreover, for every $i \in [m]$:

$$\mathsf{size}(A_i^{\mathsf{inner}}) \leq \mathsf{size}(A_i^{\mathsf{outer}}) + \mathsf{poly}(T),\ \mathsf{depth}(A_i^{\mathsf{inner}}) \leq \mathsf{depth}(A_i^{\mathsf{outer}}) + \mathsf{polylog}(T)$$
$$\mathsf{size}(\phi_i^{\mathsf{inner}}) \leq \mathsf{size}(\phi_i^{\mathsf{outer}}) + \mathsf{poly}(T),\ \mathsf{depth}(\phi_i^{\mathsf{inner}}) \leq \mathsf{depth}(\phi_i^{\mathsf{outer}}) + \mathsf{poly}(\kappa, \log T)$$

Suppose the output set B^{inner} of surviving indices in the inner protocol is μ^{inner}-succinct , where $\mu^{\mathsf{inner}} \leq T$, $\mathsf{size}(B^{\mathsf{inner}}) \leq \mathsf{poly}(T)$ and depth $\mathsf{depth}(B^{\mathsf{inner}}) \leq \mathsf{polylog}(T)$. Finally, suppose that the output predicates $(\psi_i^{\mathsf{inner}})_{i\in[m]}$ are LDE predicates (and are thus fully succinct).

The complexities of the outer honing protocol are then:

- *communication complexity* $\mathsf{cc}^{\mathsf{outer}} = \mathsf{cc}^{\mathsf{inner}} + m \cdot S \cdot \mathsf{poly}(\kappa, k, d, \log T)$.
- *round complexity* $r^{\mathsf{outer}} = r^{\mathsf{inner}} + \mathsf{polylog}(T)$.
- *prover runtime* $\mathcal{P}\mathsf{time}^{\mathsf{outer}} = \mathcal{P}\mathsf{time}^{\mathsf{inner}} + \mathsf{poly}(T, m, \kappa, \rho^{\mathsf{outer}})$.
- *verifier runtime* $\mathcal{V}\mathsf{time}^{\mathsf{outer}} = \mathcal{V}\mathsf{time}^{\mathsf{inner}} + m \cdot (S + \rho^{\mathsf{outer}} + \mu^{\mathsf{inner}}) \cdot \mathsf{poly}(\kappa, k, d, \log T))$.
- *the set B^{outer} of surviving indices is $q^{\mathsf{outer}} = (q^{\mathsf{inner}} + \mathsf{poly}(k, \log T))$-succinct where:*

  $$\mathsf{size}(B^{\mathsf{outer}}) \leq \mathsf{size}(B^{\mathsf{inner}}) + \mathsf{poly}(|U|, k, \log T)),\ \mathsf{depth}(B^{\mathsf{outer}}) \leq \mathsf{depth}(B^{\mathsf{inner}}) + \mathsf{polylog}(T)).$$

- *the predicates $(\psi_i^{\mathsf{outer}})_{i\in[m]}$ are LDE predicates (and thus fully succinct).*

Proof. The parameters of the recursive call are as in the lemma statement: for the size of U'^{expd}, recall that if A^{samp} has density larger than $1/k$ then the verifier accepts immediately in Step 2, so $|U'| \leq |U| \cdot (1 + 1/k)$. To bound the succinctness of the inner predicates ϕ_i^{inner} and sets A_i^{inner} observe that, by construction, the description of each ϕ_i^{inner} includes the description for the outer predicate, the checksums for the state matrices and the hash root matrices (and the hash tree key). The descriptions of each inner set A_i^{inner} includes the description of the outer set and the $10k \log\log(T)$-wise independent function. The sizes and depths of the predicates and sets are by construction (note that the inner predicates need to compute the hash tree over the tableau, which results in increased depth).

This accounts for the contributions of the recursive calls to the complexities. The further overheads come from:

- The checksums and UOWHF commitment use $(S + \mathsf{poly}(\kappa)) \cdot m \cdot \mathsf{poly}(d, \log T)$ communication and verfier runtime, a constant number of rounds, and $m \cdot \mathsf{poly}(T, \kappa)$ prover runtime.
- The HIAs of Step 4 add $m \cdot (\rho^{\mathsf{outer}} + \mu^{\mathsf{inner}}) \cdot \mathsf{poly}(\kappa, \log(T))$ communication and verifier runtime, $\mathsf{polylog}(T)$ rounds, and $\mathsf{poly}(T)$ prover runtime (see Theorem 3.24, note that ψ_i^{ctrt} are fully succinct). The succinctness only comes into play in letting the prover and verifier restrict their attention in the i-th HIA to the appropriate subset of rows: the subset B_i^{ctrt}, which can be computed from the intersection of A_i^{outer} and B^{inner} is $(\rho^{\mathsf{outer}} + \mu^{\mathsf{inner}})$-succinct.
- The nested IAPs of Step 5 add $(m \cdot d \cdot \mathsf{poly}(\kappa, \log(T)))$ communication, $(m \cdot (|x| + S + \rho^{\mathsf{outer}} + \mu^{\mathsf{inner}}) \cdot d \cdot \mathsf{poly}(\kappa, \log(T)))$ verifier runtime, $\mathsf{polylog}(T)$ rounds and $\mathsf{poly}(T)$ prover runtime (see Theorem 3.32, the nested sets are ρ^{outer}-succinct and the predicates, which need to know the description of B_i^{ctrt}, are μ^{inner}-succinct). The IAP's output set B^{IAP} of surviving indices and its output predicates ψ_i^{IAP} are fully succinct.

- The final holographic argument of Step 6 adds $m \cdot (\rho^{\mathsf{outer}} + \mu^{\mathsf{inner}}) \cdot \mathsf{poly}(\kappa, \log(T))$ communication and verifier runtime, $\mathsf{polylog}(T)$ rounds, and $\mathsf{poly}(T)$ prover runtime (see Theorem 3.24, note that $\psi_i^{\mathsf{samp-ctrt}}$ and ψ_i^{IAP} are fully succinct). Here too succinctness only comes into play in letting the prover and verifier restrict their attention in the i-th holographic argument to B_i^{outer}.

The set B of surviving indices for the honing protocol is derived from: (i) the set A^{samp}, (ii) the honing protocol's surviving set B^{inner}, and (iii) the nested IAP's surviving set B^{IAP}. Thus, it is $(\mu^{\mathsf{inner}} + \mathsf{poly}(k, \log T))$-succinct. As claimed, the circuit computing the set B only requires an additive overhead of $\mathsf{poly}(|U|, \log(T))$ size and $\mathsf{polylog}(T)$ depth over computing B^{inner}. The predicates ψ_i^{outer}, which are produced by the final holographic argument, are fully succinct (these are LDE predicates over the tableaux). □

Lemma 4.3 (Recursive construction soundness)**.** *Suppose that* Π^{inner} *is a* $(\eta^{\mathsf{inner}}, \delta_{ABS}^{\mathsf{inner}}, \ell)$*-honing protocol with soundness error* $\varepsilon_s(\kappa)$ *against adversaries of size* $\mathrm{s}_{\mathcal{A}}(\kappa)$ *(see Definition 3.35), where* $\eta_{\mathsf{inner}} \leq \frac{1}{100k^2 \log T \log\log T}$, $\delta_{ABS}^{\mathsf{inner}} \leq d \cdot k$ *and that* HIA *and* IAP *have soundness error* $\varepsilon^{\mathsf{hol}}, \varepsilon^{\mathsf{IAP}} = 1/\mathsf{polylog}(T)$ *against adversaries of size* $\mathrm{s}_{\mathcal{A}}(\kappa)$*. Suppose also that the UOWHF tree commitment scheme is* $(\mathrm{s}_{\mathcal{A}}(\kappa), \varepsilon_{\mathsf{UOWHF}}(\kappa))$*-secure (see Definition 3.7).*

Then Π^{outer} *is a* $(\eta^{\mathsf{outer}}, \delta_{ABS}^{\mathsf{outer}}, k \cdot \ell)$*-honing protocol, where:*

- $\eta^{\mathsf{outer}} = \frac{\eta^{\mathsf{inner}}}{2} + O\left(\frac{\log\log T}{d}\right)$.
- $\delta_{ABS}^{\mathsf{outer}} = \frac{\delta_{ABS}^{\mathsf{inner}}}{k} + d$.
- *for adversaries of size* $(\mathrm{s}_{\mathcal{A}}(\kappa) - \mathsf{poly}(T, m, \kappa))$*, the soundness error is* $\left(\varepsilon_s(\kappa) + (\varepsilon_{\mathsf{UOWHF}}(\kappa) \cdot |U| \cdot m) + \frac{1}{2\log T}\right)$.

Proof of Lemma 4.3. We begin by bounding the honing parameter. The heart, namely bounding the soundness error, follows below.

Honing parameter. For each $i \in [m]$, the set of surviving rows is $B_i^{\mathsf{outer}} = B_i^{\mathsf{IAP}} \cup B_i^{\mathsf{samp-ctrt}}$. Since the IAP executions are nested (see Definition 3.31), there is a set $B^{\mathsf{IAP}} \subset [|U|]$ of surviving rows where for each $i \in [m]$: $B_i^{\mathsf{IAP}} = A_i^{\mathsf{outer}} \cap B^{\mathsf{IAP}}$. By the parameters chosen for the IAP in Step 5 and by Theorem 3.32, each row in $[|U|]$ survives with probability at most $O(\log\log(T)/d)$. The sets $B_i^{\mathsf{samp-ctrt}}$ are formed as follows. First, the sampling step chooses the set A^{samp} where each row survives with probability $1/2k$. Then, in the recursive protocol call in Step 3, there is a subset of surviving rows $A^{\mathsf{inner}} \subseteq (A^{\mathsf{samp}})^{\mathsf{expd}}$ within the (expansion of) the part of U' that corresponds to A^{samp}. Each row in $(A^{\mathsf{samp}})^{\mathsf{expd}}$ survives with probability η^{inner}. By a union bound, for each k-row block of $(A^{\mathsf{samp}})^{\mathsf{expd}}$ (corresponding to a single row of A^{samp}), the probability that there is a surviving row in that block is at most $k \cdot \eta^{\mathsf{inner}}$. Let $B^{\mathsf{samp-ctrt}} \subseteq A^{\mathsf{samp}}$ be the k-fold contraction of the set of surviving rows in $(A^{\mathsf{samp}})^{\mathsf{expd}}$, so that $B_i^{\mathsf{samp-ctrt}} = A_i^{\mathsf{samp}} \cap B^{\mathsf{samp-ctrt}}$. By the above, each row in A^{samp} survives in $B^{\mathsf{samp-ctrt}}$ is at most $k \cdot \eta^{\mathsf{inner}}$. Putting this together, each row in $[U]$ survives in $B^{\mathsf{samp-ctrt}}$ w.p. at most $(1/2k) \cdot (k \cdot \eta^{\mathsf{inner}}) = \eta^{\mathsf{inner}}/2$. Overall, the "global" set of surviving rows is $B^{\mathsf{IAP}} \cup B^{\mathsf{samp-ctrt}}$. By a Union Bound, the probability that each row of $[U]$ survives to be included in this set is at most $(\eta^{\mathsf{inner}}/2 + O(\log\log T/d))$.

Bounding the soundness error. We proceed with the proof of soundness, building up to Lemma 4.17 below. First, note that by the parameters chosen in the lemma, it holds that:

$$\delta_{\mathsf{ABS}}^{\mathsf{inner}} \leq k \cdot \min\left(\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d, d\right). \tag{3}$$

This fact is used in the proof of Lemma 4.17. By that lemma, and accounting also for a $1/\mathsf{polylog}(T)$ probability that the verifier terminates and accepts in Step 2, the soundness error is bounded by:

$$\frac{1}{\mathsf{polylog}(T)} + \varepsilon_s(\kappa) + (\varepsilon_{\mathsf{UOWHF}}(\kappa) \cdot |U| \cdot m) + \varepsilon^{\mathsf{IAP}} + 2\varepsilon^{\mathsf{hol}} + \frac{1}{10\log T} + \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T \leq$$
$$\varepsilon_s(\kappa) + (\varepsilon_{\mathsf{UOWHF}}(\kappa) \cdot |U| \cdot m) + \frac{1}{2\log T},$$

where the last inequality follows by the bound on η^{inner} in the Lemma statement and by the choice of soundness parameters for the HIAs and the IAPs.

Notation and preliminaries. Following the notation in the overview, for each $i \in [m]$ define:

- $\widetilde{U}_i$: the matrix closest in row-distance to $U|_{A_i^{\mathsf{outer}}}$ that has checksum $\widetilde{C_{U\,i}}$.
- $R(\widetilde{U}_i)$: the matrix of hash tree roots for the encodings of the tableaux of $\widetilde{U}_i$.
- $\widetilde{R}_i$: the matrix closest in row-distance to $R(\widetilde{U}_i)$ with checksum $\widetilde{C_{R\,i}}$.
- $\widetilde{U}_i^{\mathsf{samp}}$: the closest matrix in row-distance to $U|_{A_i^{\mathsf{samp}}}$ with checksum $\widetilde{C_{U\,i}}^{\mathsf{samp}}$.
- $R(\widetilde{U}_i^{\mathsf{samp}})$: the matrix of hash tree roots for the encoding of the tableaux of $\widetilde{U}_i^{\mathsf{samp}}$.
- $\widetilde{R}_i^{\mathsf{samp}}$: the closest matrix in row-distance to $R(\widetilde{U}_i^{\mathsf{samp}})$ with checksum $\widetilde{C_{R\,i}}^{\mathsf{samp}}$.

By the properties of the checksum, for each of these definitions there can be at most one matrix with the given checksum at distance d from the appropriate matrix. Unless we explicitly note otherwise, we assume all of these matrices exist (i.e. that the checksum values are realizable). Otherwise, we syntactically define the appropriate matrix to be at infinite distance from the matrix it is supposed to be close to. We index the rows of $\widetilde{U}_i, R(\widetilde{U}_i), \widetilde{R}_i$ using the set A_i^{outer} in the natural way. Similarly, we index the rows of $\widetilde{U}_i^{\mathsf{samp}}, R(\widetilde{U}_i^{\mathsf{samp}}), \widetilde{R}_i^{\mathsf{samp}}$ using the set A_i^{samp} in the natural way.

For $i \in [m]$, we say that a row in $\widetilde{U}_i$ is "corrupted" if the value in that same row in $\widetilde{R}_i$ is not the correct hash tree root for (the tree built on) the encoding of the tableaux of $\widetilde{U}_i$, i.e. if $R(\widetilde{U}_i)$ and $\widetilde{R}_i$ differ in that row. We define the event:

$$\mathsf{E}_i^{\mathsf{corrupt}}(A \subseteq A_i^{\mathsf{outer}}) : \mathbb{1}\left\{\text{one of the rows in the set } A \text{ is corrupted}\right\}. \tag{4}$$

Note that if this event *doesn't* occur then the hash roots should be binding for all rows in the set A. We syntactically define the event $\mathsf{E}_{m+1}^{\mathsf{corrupt}}$ to be false (on any set).

Setting the table. Suppose the i-th input predicate is $\delta_{\mathsf{ABS}}^{\mathsf{outer}}$-far from satisfying the (tableaux of the) states in A_i^{outer}. The crux of the soundness proof is showing that w.h.p. after the recursive protocol call, for some $i' \geq i$, one of three "good" (for the verifier) outcomes occur:

- **Option I: cheating on the sample.** The prover's claim about $B_{i'}^{\mathsf{samp-ctrt}}$ (a subset of the sampled set) is false. This is a small set, so the verifier has made progress.
- **Option II: bound to a false claim**: the rows of the set $B_{i'}^{\mathsf{ctrt}}$ are all non-corrupted (and so the commitment is binding) *AND* the prover made a false claim about the tableaux of rows in this set. While here the set isn't small, this is progress for the verifier because it can use the binding property to derive a false claim about a much smaller set (see the overview).

  For $i \in [m]$ we define the event:

$$\mathsf{E}_i^{\mathsf{binding-cheat}} = \neg \mathsf{E}_i^{\mathsf{corrupt}}(B_i^{\mathsf{ctrt}}) \bigwedge \left(\psi_i^{\mathsf{ctrt}} \left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}},(k\cdot\ell)}} \right) = 0 \right). \tag{5}$$

- **Option III: cheating on a checksum**. Lastly, the verifier can also make progress if the checksums sent are themselves inconsistent. We distinguish between inconsistencies before and after the verifier chooses its sub-sampled sets.

  After Step (1), the i-th checksums are inconsistent if $U|_{A_i^{\mathsf{outer}}}$ or $R(\widetilde{U}_i)$ are far from being consistent with their respective checksums, or if they are inconsistent with a checksum sent for a set $j \geq i$: since the sets are nested, the matrix $\widetilde{U}_i$ induces values for the checksums $(\widetilde{C_U})_{j\in[i,m]}$, and similarly for $\widetilde{R}_i$. For $i \in [m]$ we define the event:

$$\begin{aligned}
\mathsf{E}_i^{\mathsf{chksum-cheat-1}} =& \mathbb{1}\left\{ \Delta_{\mathsf{ABS}}^{\mathsf{row}} \left(U|_{A_i^{\mathsf{outer}}}, \widetilde{U}_i \right) > d \right\} \\
& \bigvee \mathbb{1}\left\{ \Delta_{\mathsf{ABS}}^{\mathsf{row}} \left(R(\widetilde{U}_i), \widetilde{R}_i \right) > d \right\} \\
& \bigvee_{j\in[i,m]} \mathbb{1}\left\{ \text{checksum } \widetilde{C_{U}}_j \text{ is not consistent with } \widetilde{U}_i|_{A_j^{\mathsf{outer}}} \right\}, \\
& \bigvee_{j\in[i,m]} \mathbb{1}\left\{ \text{checksum } \widetilde{C_{R}}_j \text{ is not consistent with } \widetilde{R}_i|_{A_j^{\mathsf{outer}}} \right\}.
\end{aligned} \tag{6}$$

  After Step (2), the prover is cheating on the i-th checksums if they are inconsistent with the checksums for the sampled sets for any set $j \in [i, m]$: For $i \in [m]$ we define the event:

$$\begin{aligned}
\mathsf{E}_i^{\mathsf{chksum-cheat-2}} =& \bigvee_{j\in[i,m]} \mathbb{1}\left\{ \text{checksum } \widetilde{C_{U}}_j^{\mathsf{samp}} \text{ is not consistent with } \widetilde{U}_i|_{A_j^{\mathsf{samp}}} \right\} \\
& \bigvee_{j\in[i,m]} \mathbb{1}\left\{ \text{checksum } \widetilde{C_{R}}_j^{\mathsf{samp}} \text{ is not consistent with } \widetilde{R}_i|_{A_j^{\mathsf{samp}}} \right\}.
\end{aligned} \tag{7}$$

  We say that the prover is cheating on the i-th checksum if either of these events occurs:

$$\mathsf{E}_i^{\mathsf{chksum-cheat}} = \mathsf{E}_i^{\mathsf{chksum-cheat-1}} \bigvee \mathsf{E}_i^{\mathsf{chksum-cheat-2}}, \tag{8}$$

where we syntactically define the events $\mathsf{E}_0^{\mathsf{chksum-cheat}}, \mathsf{E}_{m+1}^{\mathsf{chksum-cheat}}$ to be false.

In this case, the cheating claim allows the verifier to directly hone its search to a smaller set using an IAP (see the overview).

The following fact about the checksum events follows immediately from the construction and the nesting structure of the sets, and will be helpful throughout the analysis:

Fact 4.4. *For every $i \in [1, m]$, if $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ does* not *occur, then for every $j \in [i, m]$ it holds that: $\widetilde{U}_j = \widetilde{U}_i|_{A_j^{\mathsf{outer}}}$, $\widetilde{R}_j = \widetilde{R}_i|_{A_j^{\mathsf{outer}}}$, and the event $\mathsf{E}_j^{\mathsf{chksum-cheat}}$ also does not occur.*

Breaking the recursive call. Our goal is showing that, assuming that the recursive call is sound, one of the aforementioned "good" outcomes occurs. This is shown in Proposition 4.5 and Claim 4.6. Towards this we define an event that indicates that the prover broke the recursive (inner) protocol's soundness on the i-th set. When we say that the prover "breaks" the recusions, we define this w.r.t. a particular input table. For $i \in [m+1]$ we define:

$$\widetilde{V}_i = \begin{pmatrix} \widetilde{U}_i \\ U|_{A_{i-1}^{\mathsf{samp}}} \end{pmatrix} \tag{9}$$

($\widetilde{U}_0^{\mathsf{samp}}$ and $\widetilde{U}_{m+1}$ are empty matrices). I.e. the matrix $\widetilde{V}_i$ has two parts: the top part takes values by $\widetilde{U}_i$ and the bottom part takes values by U (the original input table). Similarly to the matrix U' in the protocol, we index the rows of $\widetilde{V}_i$ using the set $(A_i^{\mathsf{outer}} \cup A_{i-1}^{\mathsf{samp}})$ in the natural way. Note, however, that the same row can appear in both parts of $\widetilde{V}_i$, and can take different values. Intuitively, if the hash roots in $\widetilde{R}_i$ are uncorrupted, then the cheating prover is "committed" to $\widetilde{U}_i$. Even if there are only a few corruptions, w.h.p. after the recursive call the prover will still be committed to tableaux for which the verifier has derived a false claim (see the overview and above). On the other hand, if there are corruptions in $\widetilde{U}_{i-1}^{\mathsf{samp}}$, then the recursive call will result in a false claim about an even smaller subset of the rows of U (here we do not need a binding commtiment, but we do need the claim to be false about the rows of the actual input U).[8]

The soundness property of the recursive call is w.r.t. a single global input table that is fixed before that protocol begins, whereas the $(\widetilde{V}_i)_{i\in[m+1]}$ tables might not be consistent with a single global table (because the $\widetilde{U}_i$ parts might not be consistent). However, if $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ doesn't occur, then $(\widetilde{U}_j)_{j\geq i}$ are all consistent with the single input table $\widetilde{U}_i$, and we define breaking the recursion on the i-th set as breaking it w.r.t. a global input table that agrees with $\widetilde{V}_i$ on its row set (the values of the rows outside $\widetilde{V}_i$ are immaterial).

$$\begin{aligned}\mathsf{E}_i^{\mathsf{recurse-break}} &= \mathbb{1}\left\{\mathcal{A}^{\mathsf{outer}} \text{ breaks soundness of } \Pi^{\mathsf{inner}} \text{ on the } i\text{-th set w.r.t. input table } \widetilde{V}_i\right\} \\ &= \mathbb{1}\left\{\left(\Delta_{\mathsf{ABS}}^{\mathsf{row}}\left(\widetilde{V}_i^{\mathsf{expd}}, \left\{Z : \phi_i^{\mathsf{inner}}\left(\mathcal{T}_{Z,\ell}\right) = 1\right\}\right) \geq \delta_{\mathsf{ABS}}^{\mathsf{inner}}\right) \bigwedge \left(\forall j \in [i, m+1] : \psi_j^{\mathsf{inner}}\left(\mathcal{T}_{\widetilde{V}_i^{\mathsf{expd}}|_{B_j^{\mathsf{inner}}},\ell}\right) = 1\right)\right\}.\end{aligned} \tag{10}$$

[8] As shown below, the adversary has some latitude to decide which of two types of false claims ensues.

Intuitively, if $\mathsf{E}_i^{\mathsf{recurse-break}}$ happens with noticeable probability, then we can build an adversary that breaks the recursive protocol on input $\widetilde{V}_i$ with similar probability (see below).

Soundness from Step (3). We show that if the rows of the i-th set are far from satisfying their predicate, then unless the prover breaks soundness on the recursive call, either it is cheating on one of the checksums, or for some set $j \in [i, m+1]$, in the j-th output of the recursive call, the corresponding (tableaux of the) rows of that output set do not satisfy the corresponding predicate *AND* the hash roots are uncorrupted (and thus binding).

Proposition 4.5. *For every $i \in [m]$, if $\Delta_{ABS}^{row}\left(U, \left\{W : \phi_i^{\mathsf{outer}}\left(\mathcal{T}_{W|_{A_i^{\mathsf{outer}}},(k\cdot\ell)}\right) = 1\right\}\right) \geq \delta_{ABS}^{\mathsf{outer}}$, then the prover's messages in Step (1) define an index $i^* \in [i, m+1]$ and:*

$$\Pr\Bigg[\neg\mathsf{E}_i^{\mathsf{chksum-cheat}} \bigwedge \neg\mathsf{E}_{i^*}^{\mathsf{recurse-break}} \bigwedge \forall j \in [i^*, m+1] : \left(\psi_j^{\mathsf{inner}}\left(\mathcal{T}_{\widetilde{V}_j^{\mathsf{expd}},\ell}\right) = 1\right) \bigvee \mathsf{E}_j^{\mathsf{corrupt}}(B_j^{\mathsf{ctrt}})\Bigg] \leq \frac{1}{10\log T} + \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T.$$

Moreover, i^* is computable in time $\mathsf{poly}(T, m, \kappa)$ from the messages sent in Step (1).

We remark that the fact that i^* is efficiently computable (from messages that are sent before the recursive call is made) is important towards allowing us to use an efficient adversary that breaks the recursive call to construct an *efficient* adversary for the inner / recursive protocol.

We defer the proof of Proposition 4.5 (see below). The following claim states that if the (tableaux of the) i-th output set of the recursive call do not satisfy the corresponding output predicate, then one of the predicates ψ_i^{ctrt} or $\psi_{i-1}^{\mathsf{samp-ctrt}}$ won't accept the (tableaux of the) rows of its set.

Claim 4.6. *For every $i \in [m+1]$:*

$$\left(\psi_i^{\mathsf{inner}}\left(\mathcal{T}_{\widetilde{V}_i^{\mathsf{expd}},\ell}\right) = 0\right) \Rightarrow \left(\psi_i^{\mathsf{ctrt}}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}},k\cdot\ell}\right) = 0\right) \bigvee \left(\psi_{i-1}^{\mathsf{samp-ctrt}}\left(\mathcal{T}_{U|_{B_{i-1}^{\mathsf{samp-ctrt}}},k\cdot\ell}\right) = 0\right),$$

where the predicates $\psi_{m+1}^{\mathsf{ctrt}}$ and $\psi_0^{\mathsf{samp-ctrt}}$ are syntactically defined to always accept their inputs.

Proof of Claim 4.6. The prover splits the LDE-predicate ψ_i^{inner} on the (tableaux of) $\widetilde{V}_i$ into LDE-predicates: (*i*) ψ_i^{ctrt} on the on the length-$(k \cdot \ell)$ tableaux of $\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}}$, and (*ii*) $\psi_{i-1}^{\mathsf{samp-ctrt}}$ on the length-$(k \cdot \ell)$ tableaux of $U|_{B_{i-1}^{\mathsf{samp-ctrt}}}$ (see Proposition 3.20). If the initial LDE-predicate is false on its input, then one of the two resulting predicates must be false on its input (we remark that a cheating prover has latitude to choose which of the resulting claims is false). □

Soundness from the HIA. If the predicate ψ_i^{ctrt} doesn't accept (the tableaux of) $\widetilde{U}_i$, then either the cheating prover breaks the HIA in Step 4, or the HIA output claim is false for $\widetilde{U}_i$:

Claim 4.7. *For every cheating prover strategy $\mathcal{A}$ for Π^{outer}, if the HIA is $\varepsilon^{\mathsf{hol}}$-sound against adversaries of size $|\mathcal{A}|$, then for every $i \in [m]$:*

$$\left(\psi_i^{\mathsf{ctrt}}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}},k\cdot\ell}\right) = 0\right) \Rightarrow \Pr\left[\mathsf{LDE}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}},k\cdot\ell}\right)\left[z_i^{\mathsf{hol}}\right] \neq v_i^{\mathsf{hol}}\right] \leq \varepsilon^{\mathsf{hol}}$$

Distance for the consistency IAP. We show that for every $i \in [m]$, if either: (i) the prover cheats on the i-th checksums (the event $\mathsf{E}_i^{\mathsf{chksum-cheat}}$), or (ii) the prover doesn't cheat on the i-th checksums, but the holographic claim is false w.r.t the (encoding of the tableaux of the) matrix $\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}}$, then the matrix M_i constructed in Step (5) of the protocol is *computationally* far from satisfying the cheating prover's claim in the IAP: it is hard to find a close matrix M_i' that satisfies the prover's claim. This latter condition is sufficient for our purposes. Towards this, we define the language checked by the IAP:

Definition 4.8. *The pair language $\mathcal{L}^{\mathsf{IAP}}$ considers:*

- *an explicit input $(k, \ell, T, 1^d, 1^\kappa, h, i, z_i^{\mathsf{hol}}, v_i^{\mathsf{hol}}, \left(\widetilde{C_{U\,i'}}, \widetilde{C_{U\,i'}}^{\mathsf{samp}}, \widetilde{C_{R\,i'}}, \widetilde{C_{R\,i'}}^{\mathsf{samp}}\right)_{i' \in [i,m]})$,*
- *an implicit input: a matrix M_i with $|A_i^{\mathsf{outer}}|$ rows as follows. The j-th row is of the form $(u_j, \rho_j, v_j, \mathsf{open}_j)$: u_j corresponds to the j-th state in $U|_{A_i^{\mathsf{outer}}}$, ρ_j corresponds to the hash root for the encoded tableau starting at u_j, v_j corresponds to the value of the z-th coordinate of the encoded tableau, and open_j should correspond to the opening of this value w.r.t. the hash root ρ_j (this is all as in the construction of the matrix M_i in Step (5) of the protocol).*

The matrix is in the language if the following all hold:

1. *the (u_j) values are consistent with the checksums $(\widetilde{C_{U\,i'}}, \widetilde{C_{U\,i'}}^{\mathsf{samp}})_{i' \in [i,m]}$,*
2. *the (ρ_j) values are consistent with the checksums $(\widetilde{C_{R\,i'}}, \widetilde{C_{R\,i'}}^{\mathsf{samp}})_{i' \in [i,m]}$,*
3. *every opening open_j is a valid opening for the value v_j w.r.t. the hash root ρ_j and location z_i,*
4. *the appropriate linear combination of the values $(v_{i,j})$ equals v_i^{hol}.*

For $i \in [m]$, the pair language $\mathcal{L}_i^{\mathsf{IAP}}$ is derived from $\mathcal{L}^{\mathsf{IAP}}$ by fixing the value of i.

For a matrix W_i with $|A_i^{\mathsf{outer}}|$ rows, we define $M(W_i)$ to be the "honestly constructed" matrix where the j-th row contains: the j-th state in W_i, the hash root of its encoded tableau, and the true values and openings for that encoding in the z-th coordinate. Thus in the protocol $M_i = M(U|_{A_i^{\mathsf{outer}}})$.

Fact 4.9 (Computing M)**.** *For any table W of states, computation length ℓ^{outer}, and coordinate z in the encoded tableau, the matrix $M(W)$ defined above can be computed in time $|W| \cdot \mathsf{poly}(\ell^{\mathsf{outer}}, \kappa)$.*

The following claim will be helpful in analyzing the soundness of the IAP call.

Claim 4.10. *Suppose M' differs from $M(U|_{A_i^{\mathsf{outer}}})$ in at most d rows. If the state-column of M' is consistent with $\widetilde{C_{U\,i}}$, then it is identical to $\widetilde{U}_i$. If it is also true that the root-column of M' is consistent with $\widetilde{C_{R\,i}}$, then it is identical to $\widetilde{R}_i$ and differs from $R(\widetilde{U}_i)$ in at most d rows.*

Proof of Claim 4.10. If M' differs from $M(U|_{A_i^{\mathsf{outer}}})$ in at most d rows then its state column can only differ from $U|_{A_i^{\mathsf{outer}}}$ in at most d rows, and if its checksum is $\widetilde{C_{U\,i}}$ then, by the properties of the checksum, it must be identical to $\widetilde{U}_i$.

If it also holds that the root column of M' has checksum $\widetilde{C_{R\,i}}$, then let J_1 be the set of rows on which $M(U|_{A_i^{\mathsf{outer}}})$ and $M(\widetilde{U}_i)$ are identical and M' differs from both of them. Let J_2 be the set of rows in which $M(U|_{A_i^{\mathsf{outer}}})$ and $M(\widetilde{U}_i)$ differ and M' differs from $M(\widetilde{U}_i)$. By construction, these two sets are disjoint and the row-distance between $M(\widetilde{U}_i)$ and M' is exactly $|J_1| + |J_2|$.

We know that M' differs from $M(U|_{A_i^{\mathsf{outer}}})$ on every row in J_1. Further, since M' has checksum $\widetilde{C_{Ui}}$, the u_j values in $\widetilde{U}_i$ must be *identical* to those in M' (see above). Thus, M' and $M(U|_{A_i^{\mathsf{outer}}})$ must differ on every row in $J_1 \cup J_2$. Since M' differs from $M(U|_{A_i^{\mathsf{outer}}})$ in at most d rows it must be the case that $|J_1| + |J_2| \leq d$. Since the roots column of M' is at distance d from $M(\widetilde{U}_i)$ and has checksum $\widetilde{C_{Ri}}$, it must equal $\widetilde{R}_i$. □

In what follows we fix the explicit input (except for i) and use $\mathcal{L}^{\mathsf{IAP}}$ to refer to the set of implicit inputs that are in the pair language with the i-th explicit input (i will be clear from the context).

Claim 4.11. *For every $i \in [m]$, if $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ occurs then $\Delta^{\mathit{row}}_{\mathit{ABS}}\left(M(U|_{A_i^{\mathsf{outer}}}), \mathcal{L}^{\mathsf{IAP}}\right) \geq d$.*

Proof. Recall the definition of $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ (Equations (6), (7), (8)). If the event holds, then:

- if $\Delta^{\mathsf{row}}_{\mathsf{ABS}}\left(U|_{A_i^{\mathsf{outer}}}, \widetilde{U}_i\right) > d$: the state-column of $M(U|_{A_i^{\mathsf{outer}}})$ is at distance d from having checksum $\widetilde{C_{Ui}}$,
- otherwise, if $\Delta^{\mathsf{row}}_{\mathsf{ABS}}\left(R(\widetilde{U}_i), \widetilde{R}_i\right) > d$: no matrix of roots at distance d from $R(\widetilde{U}_i)$ has checksum $\widetilde{C_{Ri}}$. Suppose for contradiction that M' is d-close to $M(U|_{A_i^{\mathsf{outer}}})$ and has checksums $\widetilde{C_{Ui}}$ and $\widetilde{C_{Ri}}$. By Claim 4.10, the root-column of M' is at distance at most d from $R(\widetilde{U}_i)$: a contradiction.
- otherwise, there are unique matrices $\widetilde{U}_i$ that differs from $U|_{A_i^{\mathsf{outer}}}$ in at most d rows and agreed with $\widetilde{C_{Ui}}$, and $\widetilde{R}_i$ that differs from $R(\widetilde{U}_i)$ in at most d rows and agrees with $\widetilde{C_{Ri}}$.

  if $\exists j \in [i+1, m]$ s.t. $\widetilde{U}_i$ is not consistent with $\widetilde{C_{Uj}}$, then there is no matrix of states at row-distance d from $U|_{A_i^{\mathsf{outer}}}$ that agrees with both $\widetilde{C_{Ui}}$ and $\widetilde{C_{Uj}}$.

  if $\exists j \in [i+1, m]$ s.t. $\widetilde{R}_i$ is not consistent with $\widetilde{C_{Rj}}$: for any matrix M' that d-close to $M(U|_{A_i^{\mathsf{outer}}})$ and has checksums $\widetilde{C_{Ui}}, \widetilde{C_{Ri}}$, by Claim 4.10, the row-column of M' must equal $\widetilde{R}_i$ and cannot be consistent with the checksum $\widetilde{C_{Rj}}$, so $M' \notin \mathcal{L}^{\mathsf{IAP}}$.

  the cases where $\exists j \in [i+1, m]$ s.t. $\widetilde{U}_i$ is not consistent with $\widetilde{C_{Uj}}^{\mathsf{samp}}$ or that $\widetilde{R}_i$ is not consistent with $\widetilde{C_{Rj}}^{\mathsf{samp}}$ (the event $\mathsf{E}^{\mathsf{chksum-cheat-2}}$) are handled analogously.

□

The next claim shows that if the LDE claim is false for the tableaux of $\widetilde{U}_i$, then while there might be matrices $M' \in \mathcal{L}^{\mathsf{IAP}}$ that are close to $M(U|_{A_i^{\mathsf{outer}}})$), then even though the matrix has the same roots as $\widetilde{R}_i$, one of its values in the row-set B_i^{ctrt} is opened to a different value than in $M(\widetilde{U}_i)$. We later use this claim to argue that whenever the rows in B_i^{ctrt} are uncorrupted, i.e. the roots are binding, we can find a hash collision between the opening in M' and in $M(\widetilde{U}_i)$.

Claim 4.12. *For every $i \in [m]$, if*

$$\mathsf{LDE}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}}, k\cdot\ell}\right)\left[z_i^{\mathsf{hol}}\right] \neq v_i^{\mathsf{hol}},$$

then for any matrix $M' \in \mathcal{L}^{\mathsf{IAP}}$ *s.t.* $\Delta^{row}_{ABS}\left(M(U|_{A_i^{\mathsf{outer}}}), M'\right) \leq d$*, the root-column of* M' *equals* $\widetilde{R}_i$ *and for some row* $j \in B_i^{\mathsf{ctrt}}$ *a different* $v_{i,j}$ *value is opened successfully in* $M'[j]$ *and in* $M(\widetilde{U}_i)[j]$*.*

Proof. By Claim 4.10, the state-column of M' must equal $\widetilde{U}_i$ and the root-column must equal $\widetilde{R}_i$. Since $M' \in \mathcal{L}^{\mathsf{IAP}}$, the openings for every row, and in particular in every row in B_i^{ctrt}, must be legal. Suppose for contradiction that the values $(v'_{i,j})_{j\in B_i^{\mathsf{ctrt}}}$ opened in the rows of M' equal the values $(\widetilde{v}_{i,j})_{j\in B_i^{\mathsf{ctrt}}}$ in $M(\widetilde{U}_i)$. The appropriate linear combination of the $\widetilde{v}_{i,j}$ values does not equal v_i^{hol} (because $\mathsf{LDE}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}},k\cdot\ell}\right)\left[z_i^{\mathsf{hol}}\right] \neq v_i^{\mathsf{hol}}$). Thus, the same linear combination of the $v'_{i,j}$ values won't equal v_i^{hol}, and $M' \notin \mathcal{L}^{\mathsf{IAP}}$: a contradiction. □

Remark 4.13 (The IAP predicate)**.** *The* IAP*'s output claim is a predicate on (a subset of) the rows of* $M(U)$*. This includes claims about the root of the hash tree and an opening of a certain value. We interpret this as a predicate over the tableaux of those rows in* U*, where the low-depth predicate computes the hash tree and the appropriate opening from the encoded tableau.).*

Soundness from the consistency IAP. Building on the claims above, we show that if either: (i) the prover cheats on the i-th checksums (the event $\mathsf{E}_i^{\mathsf{chksum-cheat}}$), or (ii) the prover doesn't cheat on the i-th checksums, but the holographic claim is false w.r.t the (encoding of the tableaux of the) matrix $\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}}$, then the IAP ends with a false claim (predicate) about a small subset of the tableau of the input U. The latter case assumes that the prover cannot break the commitment.

Claim 4.14. *For every cheating prover strategy* $\mathcal{A}$ *for* Π^{outer}*, if the* IAP *is* $\varepsilon^{\mathsf{hol}}$*-sound against adversaries of size* $|\mathcal{A}|$*, then for every* $i \in [m]$

$$\Pr\left[\psi^{\mathsf{IAP}}\left(\mathcal{T}_{U|_{B_i^{\mathsf{IAP}}},k\cdot\ell}\right) = 1 \middle| \mathsf{E}_i^{\mathsf{chksum-cheat}}\right] \leq \varepsilon^{\mathsf{IAP}}.$$

Proof. By Claim 4.11, the input to the IAP (the matrix $M(U|_{A_i^{\mathsf{outer}}})$) is at absolute row-distance d from the language $\mathcal{L}^{\mathsf{IAP}}$. Since the absolute row distance for the IAP is d (see Remark 3.34), the soundness of the IAP implies that the output claim is not satisfied by the subset of rows $M(U|_{A_i^{\mathsf{outer}}})|_{B_i^{\mathsf{IAP}}}$. The claim follows because we treat the output claim of the IAP as a predicate over the tableaux of those rows, see remark 4.13. □

For the second case (the holographic claim is false) we need to define an event indicating that the adversary broke the commitment scheme. For every $i \in [m]$, if the input to the IAP of Step (5) is not d-far from the language $\mathcal{L}^{\mathsf{IAP}}$, then the prover's first message in the IAP defines an (efficiently computable) $M'_i \in \mathcal{L}^{\mathsf{IAP}}$ that is d-close to $M(U)|_{A_i^{\mathsf{outer}}}$ (or otherwise the verifier will reject w.h.p.):

$$M'_i \stackrel{\text{def}}{=} \text{the matrix guaranteed by Theorem 3.32 for the } i\text{-th IAP execution in Step (5).} \tag{11}$$

By the first-message extractability guarantee of Theorem 3.32 we know that either: (i) $M'_i \in \mathcal{L}^{\mathsf{IAP}}$ and given the prover's first message in the protocol, M'_i is computable in time $\mathsf{poly}(|M(U)|) = \mathsf{poly}(|U|, \kappa, \log(T))$, or (ii) the verifier in the IAP will reject w.h.p.

If the former happens, we say that there is a hash collision if:

$$\mathsf{E}^{\mathsf{commit-break}} = \mathbb{1}\{\exists i \in [m], j \in A_i^{\mathsf{outer}} : j\text{-th row of } M_i' \text{ and } M(\widetilde{U}_i) \text{ agree on state and its correct root, openings of } z_i^{\mathsf{hol}}\text{-th coordinate are to different values.}\} \tag{12}$$

this is a targeted collision (breaking the commitment scheme) because the states in $\widetilde{U}_i$, their encoded tableaux, and the hash roots were defined in Step (1) before the verifier sent the UOWHF key. □

Claim 4.15. *For every cheating prover strategy $\mathcal{A}$ for Π^{outer}, if the IAP is $\varepsilon^{\mathsf{hol}}$-sound against adversaries of size $|\mathcal{A}|$, then for every $i \in [m]$, if $\neg \mathsf{E}_i^{\mathsf{corrupt}}(B_i^{\mathsf{ctrt}})$ occurs and also $\mathsf{LDE}\left(\mathcal{T}_{\widetilde{U}_i|_{B_i^{\mathsf{ctrt}}}, k\cdot\ell}\right)[z_i^{\mathsf{hol}}] \neq v_i^{\mathsf{hol}}$:*

$$\Pr\left[\neg \mathsf{E}^{\mathsf{commit-break}} \bigwedge \psi_i^{\mathsf{IAP}}\left(\mathcal{T}_{U|_{B_i^{\mathsf{IAP}}}, k\cdot\ell}\right) = 1\right] \leq \varepsilon^{\mathsf{IAP}}.$$

Proof. If $M(U|_{A_i^{\mathsf{outer}}})$ is d-far from $\mathcal{L}^{\mathsf{IAP}}$ (in absolute row distance), then the claim follows similarly to the proof of Claim 4.14. Otherwise, by the first message extractability guarantee, after the prover's first message, either the verifier rejects w.h.p. or there exists a matrix $M_i' \in \mathcal{L}^{\mathsf{IAP}}$ that is d-close to $M(U|_{A_i^{\mathsf{outer}}})$. By Claim 4.12, the root-column of M_i' equals $\widetilde{R}_i$, but for some row $j \in B_i^{\mathsf{ctrt}}$, a different value is opened in $M_i'[j]$ and in $M(\widetilde{U})_i)$. Both openings are legal w.r.t their respective roots: in $M(\widetilde{U})_i)$ this is by construction, and M_i' this is because the matrix is in $\mathcal{L}^{\mathsf{IAP}}$ (which checks this condition). It remains to show that the roots in the j-th row of these two matrices are identical.

By the conditions of the claim, the event $\neg \mathsf{E}_i^{\mathsf{corrupt}}(B_i^{\mathsf{ctrt}})$ occurs, so for every row $j \in B_i^{\mathsf{ctrt}}$ the root in $\widetilde{R}_i$ is uncorrupted: the roots in $\widetilde{R}_i$ and $R(\widetilde{U}_i)$ agree on these rows. Since by the above the root-column of M_i' also equals $\widetilde{R}_i$, we have that the roots in the j-th row of M_i' and $M(\widetilde{U})_i)$ are identical, and thus the event $\mathsf{E}^{\mathsf{commit-break}}$ occurs. □

Soundness from the final holographic argument. The HIA run in Step (6) checks that both predicates ψ_i^{IAP} and $\psi_i^{\mathsf{samp-ctrt}}$ accept the tableaux of their corresponding rows (which are both subsets of A_i^{outer}). Intuitively it "combines" these two initial claims into a single LDE predicate that will be false w.h.p. if either of the initial claims was false. The following claim follows directly from Step (6) of the protocol and the soundness of the HIA.

Claim 4.16. *For every cheating prover strategy $\mathcal{A}$ for Π^{outer}, if the HIA is $\varepsilon^{\mathsf{hol}}$-sound against adversaries of size $|\mathcal{A}|$ then for every $i \in [m]$, if $\psi^{\mathsf{IAP}}\left(\mathcal{T}_{U|_{B_i^{\mathsf{IAP}}}, k\cdot\ell}\right) = 0$ or if $\psi_i^{\mathsf{samp-ctrt}}\left(\mathcal{T}_{U|_{B_i^{\mathsf{samp-ctrt}}}, k\cdot\ell}\right) = 0$:*

$$\Pr\left[\psi_i^{\mathsf{outer}}\left(\mathcal{T}_{U|_{B_i^{\mathsf{outer}}}, k\cdot\ell}\right) = 1\right] \leq \varepsilon^{\mathsf{hol}}$$

Soundness: Putting it together. The soundness of the recursive construction puts together the above claims to show that any adversary who breaks soundness must either be able to find UOWHF collisions or must break the recursive call (we ignore here the probability that the verifier accepts in Step 2, as that is a low-probability event).

Lemma 4.17. *For every cheating prover strategy $\mathcal{A}$ for Π^{outer} s.t. the soundness guarantees of the* HIA *and the* IAP *hold against adversaries of size $\mathcal{A}$, for every $i \in [m]$, if it holds that:*

$$\Delta^{row}_{ABS}\left(U, \left\{W : \phi_i^{\mathsf{outer}}\left(\mathcal{T}_{W|_{A_i^{\mathsf{outer}}}, k\cdot\ell}\right) = 1\right\}\right) \geq \delta^{\mathsf{outer}}_{ABS},$$

then for every $\alpha, \beta \in [0,1]$ if:

$$\Pr\left[\forall j \in [i,m] :\ \psi_j^{\mathsf{outer}}\left(\mathcal{T}_{U|_{B_j^{\mathsf{outer}}}, k\cdot\ell}\right) = 1\right] \geq \alpha + \beta + \varepsilon^{\mathsf{IAP}} + 2\varepsilon^{\mathsf{hol}} + \frac{1}{10\log T} + \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T,$$

then there exists either:

- *an adversary $\mathcal{A}^{\mathsf{commit}}$ of size $|\mathcal{A}| + \mathsf{poly}(T, m, \kappa)$ that breaks the commitment w.p. $\frac{\beta}{m\cdot|U|}$,*
- *or an adversary $\mathcal{A}^{\mathsf{inner}}$ of size $|\mathcal{A}| + \mathsf{poly}(T, m, \kappa)$ that breaks Π^{inner} w.p. α.*

Moreover, if $\mathcal{A}$ is a uniform machine then so are $\mathcal{A}^{\mathsf{commit}}$ and $\mathcal{A}^{\mathsf{inner}}$.

Proof of Lemma 4.17. By Proposition 4.5, the prover's messages in Step (1) define an efficiently computable index $i^* \in [i, m+1]$. After the recursive call of Step (3), with all but $(\frac{1}{10\log T} + \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T)$ probability, one of the following holds:

- either $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ occurs: in this case, by Claim 4.14, after Step (5), with all but $\varepsilon^{\mathsf{IAP}}$ probability, the IAP's output claim on the i-th set will be false. When this holds, by Claim 4.16, the output claim ψ_i^{outer} will be false with all but $\varepsilon^{\mathsf{hol}}$ probability.
- or $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ occurs: this will contribute towards constructing an adversary that breaks the recursive protocol, see below.
- or there is $j \in [i^*, m+1]$ s.t. $\neg\mathsf{E}_j^{\mathsf{corrupt}}(B_j^{\mathsf{ctrt}})$ and also $\psi_j^{\mathsf{inner}}\left(\mathcal{T}_{\tilde{V}_j^{\mathsf{expd}}, \ell}\right) = 0$.

  By Claim 4.6, this means that:

  1. either the corresponding $\psi_{j-1}^{\mathsf{samp-ctrt}}$ claim is false (this can only happen for $j > i^*$). By Claim 4.16, the output claim $\psi_{j-1}^{\mathsf{outer}}$ will be false with all but $\varepsilon^{\mathsf{hol}}$ probability.
  2. or the corresponding ψ_j^{ctrt} claim is false (this can only happen for $j < m+1$). By Claim 4.7 and Claim 4.15, with all but $\varepsilon^{\mathsf{hol}} + \varepsilon^{\mathsf{IAP}}$ probability, either the event $\mathsf{E}^{\mathsf{commit-break}}$ happens, or the IAP's output claim ψ_j^{IAP} is false. If the latter happens, then by Claim 4.16, the output claim ψ_j^{outer} will be false with all but $\varepsilon^{\mathsf{hol}}$ probability.

We conclude that with all but at most $(\varepsilon^{\mathsf{IAP}} + 2\varepsilon^{\mathsf{hol}} + (1/10\log T) + \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T)$ probability, either one of the output claims $(\psi_j^{\mathsf{outer}})_{j\in[i^*,m]}$ is false, or the event $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ occurs, or the event $\mathsf{E}^{\mathsf{commit-break}}$ occurs. If the overall probability that at least one of the latter two events occurs is less than $\alpha + \beta$ then we are done. Otherwise, either the probability of $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ (for the i^* defined after Step (1)) is at least α, or the probability of $\mathsf{E}^{\mathsf{commit-break}}$ is at least β. We handle these below:

Breaking the recursive call. If the probability of $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ is at least α we construct an adversary $\mathcal{A}^{\mathsf{inner}}$ for the recursive protocol Π^{inner} with parameters $\eta^{\mathsf{inner}}, \delta_{\mathsf{ABS}}^{\mathsf{inner}}$. $\mathcal{A}^{\mathsf{inner}}$ simulates a run of the outer protocol between $\mathcal{A}$ and the verifier. The messages sent by $\mathcal{A}$ in Step (1) define the index i^*, which is computable in time $\mathsf{poly}(T, m, \kappa)$ (see Remark 4.18). The recursive call in Step (3) defines an input $(\mathcal{M}, x, 1^S, \ell, (U')^{\mathsf{expd}}, (A_i^{\mathsf{inner}})_{i=1}^{m+1}, (\phi_i^{\mathsf{inner}})_{i=1}^{m+1})$ to the inner protocol. This is the input for which $\mathcal{A}^{\mathsf{inner}}$ breaks the recursive protocol (on index i^*). If the event $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ occurs (see Equation (10)), then the adversary successfully breaks the soundness of the inner protocol. By construction, the total success probability of $\mathcal{A}^{\mathsf{inner}}$ is α and its runtime is $|\mathcal{A}| + \mathsf{poly}(T, m, \kappa)$.

Breaking the commitment. If the probability that $\mathsf{E}^{\mathsf{commit-break}}$ happens is at least β, then we construct an adversary $\mathcal{A}^{\mathsf{commit}}$ that breaks the commitment scheme. When $\mathsf{E}^{\mathsf{commit-break}}$ happens, this means that for some $i \in [m]$, there is a row in A_i^{outer} whose correct root in $M(\widetilde{U}_i)$ is opened to two different values in $M(\widetilde{U}_i)$ and in M_i' (see Equation (12)). $\mathcal{A}^{\mathsf{commit}}$ simulates a run of Π^{outer} between the cheating prover and the verifier as follows. First, it "guesses" uniformly and at random a value $i \in [m]$ and a row $j \in A_i^{\mathsf{outer}}$ to attack. The input string to the commitment security game is the encoding of the $(k \cdot \ell)$-length tableau of that row in $\widetilde{U}_i$ (importantly, this is defined before the verifier sends the UOWHF-tree keys). The sender in the commitment game sends a commitment key h, and this is used as the key sent by the verifier in Π^{outer}. $\mathcal{A}^{\mathsf{commit}}$ then completes the simulation of Π^{outer}. It computes all messages of the cheating prover and the verifier and the matrices $M(\widetilde{U})_i$ and M_i', which are both computable in time $\mathsf{poly}(T, \kappa)$ (see Fact 4.9 and the discussion following Equation (11)). If $\mathsf{E}^{\mathsf{commit-break}}$ occurs for the value i and the row j that the adversary guessed, then it breaks the commitment scheme. This happens with probability at least $(\beta/(m \cdot |U|))$. □

Deferred proofs. We conclude with the deferred proof of Proposition 4.5.

Proof of Proposition 4.5. We analyze the protocol's soundness using several additional events. For $j \in [m]$, we take q_j to be the number of corrupted rows in $\widetilde{U}_j$. We define complement events depending on whether there are "many" or "few" corrupted rows:

$$\mathsf{E}_j^{\mathsf{many-corrupt}} = \mathbb{1}\left\{q_j \geq 10k \log\log T\right\}, \; \mathsf{E}_j^{\mathsf{few-corrupt}} = \mathbb{1}\left\{q_j < 10k \log\log T\right\}. \tag{13}$$

After the prover sends its messages in Step (1), for each row $j \in [m]$ exactly one of these complementary events occurs. The index $i^* \in [i, m+1]$ is then defined as follows:

- If $\mathsf{E}_j^{\mathsf{chksum-cheat-1}}$ holds for some $j \in [i, m]$, then, by Fact 4.4, $\mathsf{E}_i^{\mathsf{chksum-cheat-1}}$ also holds. We take $i^* = i$ and the proposition follows.
- Otherwise, i.e. if $\mathsf{E}_j^{\mathsf{chksum-cheat-1}}$ is false for all $j \in [i, m]$, then we take i^* to be the smallest index for which $\mathsf{E}_{i^*}^{\mathsf{few-corrupt}}$ holds.
- Finally, if $\mathsf{E}_j^{\mathsf{chksum-cheat-1}}$ and $\mathsf{E}_j^{\mathsf{few-corrupt}}$ are false for all $j \in [i, m]$, then we take $i^* = m+1$ (we syntactically define the event $\mathsf{E}_{m+1}^{\mathsf{few-corrupt}}$ to be true).

Remark 4.18 (Efficiently computing i^*)**.** *As claimed, i^* is efficiently computable from messages sent in Step (1): For $j \in [m]$, each of the matrices $\widetilde{U}_j$ is computable in time $\mathsf{poly}(|U|, S)$ from the respective matrix U_j (because the checksum is efficiently decodable, see Proposition 3.21). Each matrix $R(\widetilde{U}_j)$ is computable in time $\mathsf{poly}(|U|, (k \cdot \ell), \kappa)$ from $\widetilde{U}_j$ (by computing the hash root for the*

encoded computation for each state). Finally, $\widetilde{R}_j$ is computable from $R(\widetilde{U}_j)$ in in time $\mathsf{poly}(|U|,\kappa)$ (again, by decoding the checksum). These matrices allow the cheating prover to compute whether $\mathsf{E}_i^{\mathsf{chksum-cheat}-1}$ holds, and whether $\mathsf{E}_j^{\mathsf{few-corrupt}}$ or $\mathsf{E}_j^{\mathsf{many-corrupt}}$ holds for each $j \in [i,m]$. We conclude that i^* can indeed be computed in time $\mathsf{poly}(T,m,\kappa)$ (recall that we assume $T \geq \max\{|U|,(k\cdot\ell),S\}$).

We need to prove the proposition for the case that $\mathsf{E}_j^{\mathsf{chksum-cheat}-1}$ is false for all $j \in [i,m]$. By Fact 4.4, we can assume for the remainder of the proof that the single "global" matrix $\widetilde{U}_i$ is consistent with every matrix $\widetilde{U}_j$ and checksum $\widetilde{C_{U}}_j$ for $j \in [i,m]$, and that the single "global" matrix $\widetilde{R}_i$ is consistent with every matrix $\widetilde{R}_j$ and checksum $\widetilde{C_{R}}_j$ for $j \in [i,m]$. Thus, the question of whether a certain row is corrupted does not depend on the index j of the matrices being considered: either the row is corrupted in all matrices $j \in [i,m]$ or it is not.

By the choice of i^*, $\mathsf{E}_{i^*}^{\mathsf{few-corrupt}}$ holds, and so does $\mathsf{E}_j^{\mathsf{few-corrupt}}$ for every $j \in [i^*,m]$. Let $C \subseteq A_{i^*}^{\mathsf{outer}}$ be the set of corrupted rows in $A_{i^*}^{\mathsf{outer}}$. Because of the nested structure of the sets, the set of corrupted rows in each A_j^{outer} is a subset of C. Let $D \subset [|U|]$ be the (k-fold contraction of the) set of surviving rows after the recursive call of Step (3). Since the surviving rows are 1-wise uniform, the probability that each row in C survives is at most $\eta^{\mathsf{inner}} \cdot k$ (each row in the k-fold expansion survives w.p. η^{inner}). Taking a Union Bound, we can bound the probability that *there exists* a row in C that survives. When no rows in C survive, , none of the corrupted rows in any set A_j^{outer} survive:

$$\Pr\left[\exists j \in [i^*,m] :\ \mathsf{E}_j^{\mathsf{corrupt}}(B_j^{\mathsf{ctrt}})\right] \leq \eta^{\mathsf{inner}} \cdot k \cdot |C| \leq \eta^{\mathsf{inner}} \cdot 10k^2 \log\log T. \tag{14}$$

This is important because when the sets are uncorrupted their hash roots should be binding (to the corresponding rows in $\widetilde{U}_i$).

Next, we show that the i^*-th inner predicate $\phi_{i^*}^{\mathsf{inner}}$ in the recursive call is far from being satisfied by (the tableaux of) $\widetilde{V}_{i^*}$. This is shown in Claim 4.19 below. In turn, this implies that either the adversary needs to break soundness of the recursive call, or one of the outer predicates is not satisfied by the (tableaux of) the corresponding rows in the corresponding output matrix. The claim follows using the binding guarantee of Equation (14).

Claim 4.19. *Assume that $\mathsf{E}_i^{\mathsf{chksum-cheat}-1}$ does* not *hold, then:*

$$\Pr\left[\neg\mathsf{E}_i^{\mathsf{chksum-cheat}} \bigwedge \Delta_{ABS}^{row}\left(\widetilde{V}_{i^*}^{\mathsf{expd}}, \left\{Z : \phi_{i^*}^{\mathsf{inner}}\left(\mathcal{T}_{Z,\ell}\right) = 0\right\}\right) < k \cdot \min\left(\delta_{ABS}^{\mathsf{outer}} - d, d\right)\right] \leq \frac{1}{10\log T}.$$

Proof. The proof proceeds by a case analysis over the value of i^*. Before diving into the details, at a high level: if $i^* = i$, the table U is far from satisfying the outer predicate ϕ_i^{outer}. This implies that the k-fold expansion of $\widetilde{U}_i$ is far from satisfying the inner predicate ϕ_i^{inner} (which checks that the outer predicate is satisfied on the appropriate part of its input). In the case where $i^* > i$, we know that $\mathsf{E}_{i^*-1}^{\mathsf{many-corrupt}}$ holds (otherwise i^* would be smaller). This means that w.h.p. there will be a corrupted row in the sampled set $A_{i^*-1}^{\mathsf{samp}}$, which in turn means that the k-fold expansion of $\widetilde{V}_{i^*}$ will be far from satisfying the inner predicate $\phi_{i^*}^{\mathsf{inner}}$ (or the checksums sent in Step (2) will be inconsistent). We proceed with the detailed analysis of both cases:

Case *I*: $i^* = i$. We show that either the i-th checksum is corrupted, or the matrix $\widetilde{V}_i$ is far from satisfying its corresponding predicate. Under the conditions of Proposition 4.5 the i-th "outer" input is far from satisfying its predicate: $\Delta_{\mathsf{ABS}}^{\mathsf{row}}\left(U, \left\{W : \phi_i^{\mathsf{outer}}\left(\mathcal{T}_{W|_{A_i^{\mathsf{outer}}},(k\cdot\ell)}\right) = 1\right\}\right) \geq \delta_{\mathsf{ABS}}^{\mathsf{outer}}$. If

the event $\mathsf{E}_i^{\mathsf{chksum-cheat}-1}$ doesn't occur, then $\widetilde{U}_i$ differs from $U|_{A_i^{\mathsf{outer}}}$ in at most d rows. By a triangle inequality: $\Delta_{\mathsf{ABS}}^{\mathsf{row}}\left(\widetilde{U}_i, \left\{W : \phi_i^{\mathsf{outer}}\left(\mathcal{T}_{W,(k\cdot\ell)}\right) = 1\right\}\right) \geq \delta_{\mathsf{ABS}}^{\mathsf{outer}} - d$.

Since ϕ_i^{inner} checks that ϕ_i^{outer} accepts (the tableaux of) its input restricted to the corresponding rows (Step (3a)), for any matrix W at absolute row-distance $(\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d)$ from $\widetilde{U}_i$ and for any matrix W' with $|A_i^{\mathsf{samp}}|$ rows:

$$\phi_i^{\mathsf{inner}}\left(\mathcal{T}_{\begin{pmatrix} W^{\mathsf{expd}} \\ W'^{\mathsf{expd}} \end{pmatrix},\ell}\right) = 0.$$

We conclude that ϕ_i^{inner} rejects the (encoded tableaux of the) k-fold expansion of any matrix at overall row-distance $(\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d)$ from $\widetilde{V}_i$ (the values in the rows corresponding to A_i^{samp} don't matter).

Since ϕ_i^{inner} also checks "internal consistency" (Step (3b)), we conclude that $\widetilde{V}_i^{\mathsf{expd}}$ must also differ in at least $k \cdot (\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d)$ rows from any Z that makes ϕ_i^{inner} accept: suppose that there is a Z at the aforementioned row-distance from $\widetilde{V}_i^{\mathsf{expd}}$ that makes ϕ_i^{inner} accept. Because of the internal consistency test, this means that if we take Z^{ctrt} to be the k-fold contraction of Z (containing the first entry from each k-entry block of Z), then the k-fold expansion of Z^{ctrt} equals Z. For every row j in which Z^{ctrt} differs from $\widetilde{V}_i$, their k-fold expansions differ in every state in the j-th block. This follows by Fact 3.12 and Remark 3.13, using the fact that the j-th block of Z does not contain any corrupted states (i.e. the predecessor and successor functions always agree, otherwise ϕ_i^{inner} rejects Z). Thus, the absolute row distance between Z^{ctrt} and $\widetilde{V}_i$ is at most $(\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d)$. This means that $\phi_i^{\mathsf{inner}}((Z^{\mathsf{ctrt}})^{\mathsf{expd}}) = \phi_i^{\mathsf{inner}}(Z) = 0$: a contradiction. Indeed, even if only *one* of the two tableaux is valid, they disagree everywhere: suppose u's tableau is the valid one. By the above, the two tableau disagree in the prefix of u''s tableau where the states are uncorrupted, and they also disagree in the suffix where u''s tableau is corrupted (because u's is not). □

Case *II*: $i^* \in [i+1, m+1]$. $\mathsf{E}_{i^*-1}^{\mathsf{many-corrupt}}$ holds (otherwise i^* would be smaller). Here w.h.p. the subsampled set $A_{i^*-1}^{\mathsf{samp}}$ includes a corrupted row. Since the choices are $10k^2 \log\log T$-wise independent:

$$\begin{aligned}
\Pr\left[\mathsf{E}_{i^*-1}^{\mathsf{many-corrupt}} \bigwedge \neg\mathsf{corrupt}_{i^*-1}(A_{i^*-1}^{\mathsf{samp}})\right] &\leq \left(1 - \frac{1}{2k}\right)^{10k \log\log T} \\
&= \left(1 - \frac{1}{2k}\right)^{2k \cdot 5 \log\log T} \\
&\leq e^{-5\log\log T} \\
&\leq \frac{1}{10\log(T)}, \qquad (15)
\end{aligned}$$

where the final inequality holds for T larger than some constant.

When there is a corrupted row in the (i^*-1)-th subsampled set, the cheating prover has two bad choices: if the checksum it sends for the sampled set is *not* consistent with $\widetilde{U}_i$, then the event $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ occurs. Otherwise, $\widetilde{U}_{i^*-1}^{\mathsf{samp}}$ has a corrupted root and those rows will be rejected by $\phi_{i^*}^{\mathsf{inner}}$ (which checks the values of the roots). In turn, this also means that the rows of $U|_{A_{i^*-1}^{\mathsf{samp}}}$ are far from making that predicate accept. We formalize this in the following claim:

Subclaim 4.20.

$$\mathsf{corrupt}_{i^*-1}(A^{\mathsf{samp}}_{i^*-1}) \Rightarrow \mathsf{E}^{\mathsf{chksum-cheat}}_{i} \bigvee \Delta^{row}_{ABS}\left(\widetilde{V}^{\mathsf{expd}}_{i^*}, \left\{Z : \phi^{\mathsf{inner}}_{i^*}\left(\mathcal{T}_{Z,\ell}\right) = 1\right\}\right) \geq k \cdot d$$

Proof of Subclaim 4.20. Since we assume that the event $\mathsf{corrupt}_{i^*-1}(A^{\mathsf{samp}}_{i^*-1})$ occurs:

$$R\left(\widetilde{U}_{i^*-1}\right)|_{A^{\mathsf{samp}}_{i^*-1}} \neq \widetilde{R}_{i^*-1}|_{A^{\mathsf{samp}}_{i^*-1}}. \tag{16}$$

Assume that the event $\mathsf{E}^{\mathsf{chksum-cheat}}_{i^*-1}$ doesn't occur (otherwise we are done: by Fact 4.4, $\mathsf{E}^{\mathsf{chksum-cheat}}_{i^*-1} \Rightarrow \mathsf{E}^{\mathsf{chksum-cheat}}_{i}$). This means that there exists a matrix $\widetilde{U}^{\mathsf{samp}}_{i^*-1}$ at absolute row-distance d from $\widetilde{U}_{i^*-1}|_{A^{\mathsf{samp}}_{i^*-1}}$ with the checksum $\widetilde{C}^{\mathsf{samp}}_{U i^*-1}$ and that the matrices $\widetilde{U}_{i^*-1}|_{A^{\mathsf{samp}}_{i^*-1}}$ and $\widetilde{U}^{\mathsf{samp}}_{i^*-1}$ both have this checksum. Thus, these matrices must be identical:

$$\widetilde{U}_{i^*-1}|_{A^{\mathsf{samp}}_{i^*-1}} = \widetilde{U}^{\mathsf{samp}}_{i^*-1}. \tag{17}$$

A similar argument shows that (assuming $\mathsf{E}^{\mathsf{chksum-cheat}}_{i^*-1}$ doesn't occur):

$$\widetilde{R}_{i^*-1}|_{A^{\mathsf{samp}}_{i^*-1}} = \widetilde{R}^{\mathsf{samp}}_{i^*-1}. \tag{18}$$

By Equations (16), (17), (18) we conclude that:

$$R(\widetilde{U}^{\mathsf{samp}}_{i^*-1}) \neq \widetilde{R}^{\mathsf{samp}}_{i^*-1}. \tag{19}$$

We now argue that the k-fold expansion of the matrix

$$\widetilde{V}_{i^*} = \begin{pmatrix} \widetilde{U}_{i^*} \\ U|_{A^{\mathsf{samp}}_{i^*-1}} \end{pmatrix}$$

is far from satisfying the predicate $\phi^{\mathsf{inner}}_{i^*}$. Consider any matrix W (with $|A^{\mathsf{outer}}_{i^*}| + |A^{\mathsf{samp}}_{i^*-1}|$ rows.[9] that differs from $\widetilde{V}_{i^*}$ in at most d rows. In particular, this means that $W|_{A^{\mathsf{samp}}_{i^*-1}}$ differs from $U|_{A^{\mathsf{samp}}_{i^*-1}}$ in at most d rows. If the checksum of $W|_{A^{\mathsf{samp}}_{i^*-1}}$ does not equal $\widetilde{C}^{\mathsf{samp}}_{U i^*-1}$, then W^{expd} won't satisfy $\phi^{\mathsf{inner}}_{i^*}$ (which checks consistency with this checksum in Step (3c)). Otherwise, if the checksums are equal, then similarly to Equation (17):

$$W|_{A^{\mathsf{samp}}_{i^*-1}} = \widetilde{U}^{\mathsf{samp}}_{i^*-1}. \tag{20}$$

Putting together Equations (19) and (20) we conclude that:

$$R(W|_{A^{\mathsf{samp}}_{i^*-1}}) \neq \widetilde{R}^{\mathsf{samp}}_{i^*-1}. \tag{21}$$

Recall that $\widetilde{R}^{\mathsf{samp}}_{i^*-1}$ differs from $R(\widetilde{U}^{\mathsf{samp}}_{i*-1})$ in at most d rows. By Equation (20), we conclude that $\widetilde{R}^{\mathsf{samp}}_{i^*-1}$, also differs from $R(W|_{A^{\mathsf{samp}}_{i^*-1}})$ in at most d rows, but but they are not equal (by Equation (21)). Thus, the checksum of $R(W|_{A^{\mathsf{samp}}_{i^*-1}})$ cannot be $\widetilde{C}^{\mathsf{samp}}_{R i^*-1}$ and W^{expd} cannot satisfy $\phi^{\mathsf{inner}}_{i^*}$ (which checks this in Step (3d)).

$\widetilde{V}_{i^*}$ is at absolute row distance d from any W s.t. $\phi^{\mathsf{inner}}_{i^*}$ accepts W^{expd}. This implies that $\widetilde{V}^{\mathsf{expd}}_{i^*}$ is at absolute distance $(k \cdot d)$ from making $\phi^{\mathsf{inner}}_{i^*}$ accept due to the "internal consistency" check, Step (3b) (see the argument that concludes the proof for Case I above). □

[9] The row sets of $\widetilde{V}_{i^*}$ and of W are $A^{\mathsf{outer}}_{i^*} \coprod A^{\mathsf{samp}}_{i^*-1}$, where we abuse notation and use indices in $A^{\mathsf{outer}}_{i^*} \cup A^{\mathsf{samp}}_{i^*-1}$ in the natural way. See Definition 3.2 and the discussion that follows it.

For the soundness argument, we imagine that the adversary is trying to cheat in the recursive call when imagining that the values of the rows in the sets $(A_i^{\mathsf{inner}}, \ldots, A_{m+1}^{\mathsf{inner}})$ equal the values in $\widetilde{V}_i$. Assume that $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ doesn't occur: by Claim 4.19, w.h.p. the i^*-th set in the recursive call is at (absolute row) distance

$$k \cdot \min\left(\delta_{\mathsf{ABS}}^{\mathsf{outer}} - d, d\right) \geq \delta_{\mathsf{ABS}}^{\mathsf{inner}} \tag{22}$$

from its corresponding predicate (the inequality is by Equation (3)). Thus, unless the event $\mathsf{E}_{i^*}^{\mathsf{recurse-break}}$ occurs (see Equation (10)), for some $j \geq i^*$, the recursive call returns a false claim about the tableau of $\widetilde{V}_i|_{B_j^{\mathsf{inner}}} = \widetilde{V}_j|_{B_j^{\mathsf{inner}}}$ (the last inequality is because $\mathsf{E}_i^{\mathsf{chksum-cheat}}$ doesn't occur). Thus:

$$\mathsf{corrupt}_{i^*-1}(A_{i^*-1}^{\mathsf{samp}}) \bigwedge \neg \mathsf{E}_i^{\mathsf{chksum-cheat}} \Rightarrow \mathsf{E}_{i^*}^{\mathsf{recurse-break}} \bigvee_{j \in [i^*, m+1]} \left(\psi_j^{\mathsf{inner}} \left(\mathcal{T}_{\widetilde{V}_j^{\mathsf{expd}}|_{B_j^{\mathsf{inner}}, \ell}} \right) = 0 \right). \tag{23}$$

Further, by Equation (14), the hash roots are binding on all these sets. The claim follows by combining Equations (14), (15) and (23). □

4.3 The Resulting Honing Protocol

Theorem 4.21. *Assume there exists a one-way function secure against $\lambda(\kappa)$-size adversaries. There is a honing protocol for the multi-tableau language as follows: for an instance to that language $(\mathcal{M}, x, 1^S, \ell, U, (A_i)_{i\in[m]}, (\phi_i)_{i\in[m]})$. Take $T = 2|U| \cdot \ell$, where we require that $|x|, S, \kappa, m, \rho \leq T$, that $\mathsf{polylog}(T) \leq |U|$, and that the sets A_i and predicates ϕ_i are ρ-succinct with $\mathsf{poly}(T)$ size and $\mathsf{polylog}(T)$ depth. The protocol is public-coin with perfect completeness. For desired soundness $\varepsilon \geq \mathsf{poly}(1/\kappa)$, it is ε-sound against adversaries of size $(\lambda(\kappa)/\mathsf{poly}(T))$ and has:*

- ***Absolute row distance:*** $\delta_{ABS} = \mathsf{polylog}(T)$.
- ***Honing parameter:*** $\eta = \Theta\left(\frac{\log(1/\varepsilon)}{\log(T)}\right)$.
- ***Communication complexity:*** $\mathsf{cc} = m \cdot S \cdot \mathsf{poly}(\kappa, \log T)$.
- ***Round complexity:*** $r = \mathsf{polylog}(T)$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time} = m \cdot (|x| + S + \rho) \cdot \mathsf{poly}(\kappa, \log T)$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

The output sets $\{B_i\}_{i\in[m]}$ are $(\rho + \mathsf{polylog}(T))$-succinct, where each B_i has size $(\mathsf{size}(A_i) + \mathsf{poly}(|U|, \log(T)))$ and depth $(\mathsf{depth}(A_i) + \mathsf{polylog}(T))$. The output predicates ψ_i are LDE predicates.

Proof. The idea is to use recursion, with the protocol of Corollary 4.1 as the base of the recursion, and the recursive step using the protocol of Figure 1 (Lemmas 4.2 and 4.3). We number the layers of the recursion from 0 (the top / outermost layer) to r (the bottom / base). The recursion keeps track of parameters using a (parenthetical) superscript. For each level $j \in [r]$, we construct the protocol for parameters $(\eta^{(j-1)}, \delta_{\mathsf{ABS}}^{(j-1)})$ using the recursive construction of Figure 1, with the

protocol for level j as the inner protocol. At the base of the recursion, we use the IAP of Corollary 4.1. At the top level of the recursion $\ell^{(0)} = \ell$ is the length of each tableau, $u^{(0)} = |U|$ is the input table size and $T^{(0)} = u^{(0)} \cdot \ell^{(0)}$ is the total size of the computation. The j-th level of the recursion gives protocols for tableaux of length $\ell^{(j)} = \ell/k^j$. Taking $u^{(j)}$ to be the size of the table $U^{(j)}$ in the j-th level of the recursion, we take $T^{(j)} = u^{(j)} \cdot \ell^{(j)}$ to be the total size of the computation in the j-th level of the recursion. The computation size increases (mildly) as the recursion progresses, because add the sampled rows to the table U' before taking the k-fold expansion). We have:

$$\forall j \in [0,r] : u^{(j)} = |U^{(j)}| \leq \left(1 + \frac{1}{k}\right) \cdot k \cdot |U^{(j-1)}|,$$

$$T^{(j)} = u^{(j)} \cdot \ell^{(j)} \leq \left(1 + \frac{1}{k}\right)^j \cdot T^{(0)}$$

Looking ahead, the recursion depth will be smaller than $k/2$, so the computation size at the base of the recursion can be upper bound by T:

$$\forall j \in [0,r] : T^{(0)} \leq T^{(j)} \leq T \triangleq 2|U^{(0)}| \cdot \ell^{(0)}$$

We fix $k = \log(T)$ (the same value of k is used at all levels of the recursion), and we stop the recursion when we get to computations of length $\ell^{(r)} = \mathsf{polylog}(T)$, where we ensure that $\ell^{(r)} \geq k$ (and thus $l^{(j)} \geq k$ holds at all levels of the recursion). The depth of the recursion is $r = \lceil \log_k(\ell^{(0)}/\ell^{(r)}) \rceil = O(\log(\ell)/\log\log(T))$. Note that indeed, as we required above, $r \leq k/2$.

Parameter setting. For $j \in [0,r]$, we set the parameters as follows:

- **Checksum parameter.** $d^{(j)} = \left\lceil \left(1 + \frac{1}{k}\right) \cdot d^{(j+1)} \right\rceil$, where $d^{(r)} = 100k^2 \log^3 T = \mathsf{polylog}(T)$ (the definition for the bottom level of the recursion is syntactic: there is no checksum there, we use the IAP of Corollary 4.1).
- **Absolute distance promise.** $\delta^{(j)}_{\mathsf{ABS}} = \left(1 + \frac{1}{k}\right) d^{(j)}$.
- **Honing parameter.** $\eta^{(j)} = \max\left\{\eta^{(j+1)}, \frac{\eta^{(j+1)}}{2} + O\left(\frac{\log\log(T)}{d^{(j)}}\right)\right\}$, where $\eta^{(r)} = O\left(\frac{\log\log(T)}{d^{(r)}}\right)$

For the base of the recursion, the promise on the absolute distance is $\delta^{(r)}_{\mathsf{ABS}} \geq d^{(r)}$ and the honing parameter is $\eta^{(r)} = \left(\frac{\log\log T}{d^{(r)}}\right)$. The soundness error is $\varepsilon^{(r)} = 1/\mathsf{polylog}(T)$. This setting of parameters is achievable (for large enough T) for the IAP of Corollary 4.1.

For the other levels of the recusion, the parameter settings satisfy the conditions of Lemmas 4.2 and 4.3. Lemma 4.2 requires that $k \leq \ell^{(j)}$ (which always holds, see above) and $|x|, S, \kappa, d^{(j)}, m \leq T^{(j)}$, which all hold. For the j-th recursive call, Lemma 4.3 requires that $\eta^{(j+1)} \leq \frac{1}{100k^2 \log(T^{(j)}) \log\log(T^{(j)})}$ and $\delta^{(j+1)}_{\mathsf{ABS}} \leq d^{(j)} \cdot k$. The latter condition holds by construction. The former condition also holds—we prove by induction that for all $j \in [0,r]$:

$$\eta^{(j)} \leq \frac{\log(T)}{d^{(r)}} \tag{24}$$

This is true by construction at the base of the induction ($j = r$). For larger j, we assume by induction that $\eta^{(j+1)} \leq \frac{\log(T)}{d^{(r)}}$. Thus also:

$$\frac{\eta^{(j+1)}}{2} + O\left(\frac{\log\log T}{d^{(j)}}\right) \leq \frac{\log(T)}{2d^{(r)}} + O\left(\frac{\log\log T}{d^{(j)}}\right) \leq \frac{\log(T)}{d^{(r)}},$$

where the final inequality holds for T large enough that the $O(\log\log T)$ term is smaller than $\log(T)/2$. Thus we also have that:

$$\eta^{(j)} = \max\left\{\eta^{(j+1)}, \frac{\eta^{(j+1)}}{2} + O\left(\frac{\log\log T}{d^{(j)}}\right)\right\} \leq \frac{\log(T)}{d^{(r)}}.$$

The relationship between the parameters for the outer and inner protocols in the j-th level of the recursion is consistent with the guarantees of Lemma 4.3:

- The absolute distance guarantee is:

$$\delta^{(j)}_{\mathsf{ABS}} = d^{(j)} + \frac{d^{(j)}}{k} \geq d^{(j)} + \frac{\delta^{(j+1)}_{\mathsf{ABS}}}{k}.$$

- The honing parameter for the outer call is:

$$\frac{\eta^{(j+1)}}{2} + O\left(\frac{\log\log T}{d^{(j}}\right) \leq \eta^{(j)} \leq \frac{\log(T)}{d^{(r)}},$$

  see Equation (24).

Since $r \leq k/2$ (see above), for all levels of the recursion $d^{(j)}, \delta^{(j)}_{\mathsf{ABS}} = \mathsf{polylog}(T)$.

Finally, the number of sets for the j-th call is $m^{(j)} = (m + j)$, so the number of sets is always bounded by $(m + \log(T))$.

Soundness error. At the base of the recursion, the IAP of Corollary 4.1 has soundness error $\varepsilon^{(r)} = 1/\mathsf{polylog}(T)$ against adversaries of size $\mathrm{s}^{(r)}_{\mathcal{A}} = (\lambda(\kappa)/\mathsf{poly}(T))$. We assume that the UOWHF tree commitment guarantees that any adversary of size $\lambda(\kappa)/\mathsf{poly}(T)$ has success probability $\varepsilon_{\mathsf{UOWHF}} \leq 1/(4T \cdot m^{(r)} \cdot \log(T))$ (otherwise, there is a fixed polynomial f s.t. $T \geq f(\lambda(\kappa))$, and the soundness guarantee is vacuous).

By an inductive argument using Lemma 4.17, for $j \in [r]$, we conclude that the protocol obtained by the j-th level of the recursion is sound against adversaries of size

$$\mathrm{s}^{(j-1)}_{\mathcal{A}} = \lambda(\kappa)/\mathsf{poly}(T) - (r - j + 1) \cdot \mathsf{poly}(T, m, \kappa) = \lambda(\kappa)/\mathsf{poly}(T),$$

with soundness error at most:

$$\varepsilon^{(j-1)} = \varepsilon^{(r)} + (r - j + 1) \cdot \frac{3}{4\log T} \leq \frac{1}{2},$$

where the last inequality uses the fast that $r \leq \log(T)/2$ and $\varepsilon^{(r)} \leq 1/8$. We amplify the soundness error to $\varepsilon \geq 1/T$ using parallel repetition [BIN97, Hai13] with $O(\log(1/\varepsilon)) = O(\log(T))$ overhead.

Circuit succinctness and complexities. At level 0 of the recursion the predicates $\phi_i^{(0)} = \phi_i$ and the sets $A_i^{(0)} = A_i$ are $(\rho^{(0)} = \rho)$-succinct circuits of size $\mathsf{size}^{(0)} = \mathsf{poly}(T)$ and depth $\mathsf{depth}^{(0)} = \mathsf{polylog}(T)$. By induction, using Lemma 4.2 and the above bounds on k, m and $d^{(j)}$, the predicates and sets for the j-th recursive call are $(\rho^{(j)} = \rho + (S + \mathsf{poly}(\kappa)) \cdot \mathsf{polylog}(T))$-succinct circuits of size $\mathsf{poly}(T)$ and depths $(j \cdot \mathsf{poly}(\kappa, \log(T)))$ (for the predicates) and $(j \cdot \mathsf{polylog}(T))$ (for the sets).

The recursion depth is smaller than $\log(T)$, so we conclude that in every recursive call, the sets and predicates are $((S + \mathsf{poly}(\kappa)) \cdot \mathsf{polylog}(T))$-succinct circuits of $\mathsf{poly}(T)$ size and depths $\mathsf{poly}(\kappa, \log(T))$ (for the predicates) and $\mathsf{polylog}(T)$ (for the sets).

Similarly, using induction and Lemma 4.2, the output sets $B_i = B_i^{(0)}$ are $(\rho + \mathsf{polylog}(T))$-succinct circuits of size $\mathsf{poly}(T)$ and depth $\mathsf{polylog}(T)$.

Protocol complexities: base of the recursion. At the bottom level r with $\delta_{\mathsf{ABS}}^{(r)} = \mathsf{polylog}(T)$ and soundness error $\varepsilon^{(r)} = 1/\mathsf{polylog}(T)$, the complexities from Corollary 4.1 are:

- **Communication complexity:** $\mathsf{cc}^{(r)} = m^{(r)} \cdot \delta_{\mathsf{ABS}}^{(r)} \cdot \ell^{(r)} \cdot S \cdot \mathsf{polylog}(T) = m \cdot S \cdot \mathsf{polylog}(T)$.
- **Round complexity:** $r^{(r)} = \ell^{(r)} \cdot \mathsf{polylog}(T) = \mathsf{polylog}(T)$.
- **Verifier running time:** $\mathcal{V}\mathsf{time}^{(r)} = m^{(r)} \cdot \delta_{\mathsf{ABS}}^{(r)} \cdot (|x| + \rho^{(r)}) \cdot \mathsf{poly}(\ell^{(r)}, \log(T)) = m \cdot (|x| + S + \mathsf{poly}(\kappa) + \rho) \cdot \mathsf{polylog}(T)$.
- **Prover running time:** $\mathcal{P}\mathsf{time}^{(r)} = m^{(r)} \cdot \mathsf{poly}(T) = \mathsf{poly}(T)$.

Complexity overheads: inductive step. For each recursive level $j \in [r]$, by Lemma 4.2, the additive overhead is:

- cc **overhead:** $(S + \mathsf{poly}(\kappa)) \cdot m^{(j-1)} \cdot \mathsf{poly}(k, d^{(j-1)}, \log T) = (S + \mathsf{poly}(\kappa)) \cdot m \cdot \mathsf{polylog}(T)$.
- **Round overhead:** $\mathsf{polylog}(T)$.
- $\mathcal{V}\mathsf{time}$ **overhead:** $(|x| + S + \mathsf{poly}(\kappa) + \rho^{(j-1)}) \cdot m^{(j-1)} \cdot \mathsf{poly}(k, d^{(j-1)}, \log T) = (|x| + S + \mathsf{poly}(\kappa) + \rho) \cdot m \cdot \mathsf{polylog}(T)$.
- $\mathcal{P}\mathsf{time}$ **overhead:** $\mathsf{poly}(T, m^{(j-1)}, \kappa, \rho^{(j-1)}) = \mathsf{poly}(T)$.

Total complexity. The recursion depth is $r = O(\log T)$, so the total complexities are bounded:

- **Total communication:** $\mathsf{cc} = m \cdot (S + \mathsf{poly}(\kappa)) \cdot \mathsf{polylog}(T)$.
- **Total rounds:** $r = \mathsf{polylog}(T)$.
- **Total verifier time:** $\mathcal{V}\mathsf{time} = m \cdot (|x| + S + \mathsf{poly}(\kappa) + \rho) \cdot \mathsf{polylog}(T)$.
- **Total prover time:** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

□

4.4 Arguments for Bounded Space

Theorem 4.22 (Formal statement of Theorem 1.2)**.** *Assume there exists a one-way function secure against $\lambda(\kappa)$-size adversaries. There is an argument system for space-$S = S(n)$ time-$T = T(n) \geq n$ computations, where we require that $\kappa \leq T$. The protocol is public-coin with perfect completeness. For desired soundness $\varepsilon \geq \mathsf{poly}(1/\kappa)$ it is ε-sound against adversaries of size $\lambda(\kappa)/\mathsf{poly}(T)$ and has:*

- ***Communication complexity:*** $\mathsf{cc} = S \cdot \mathsf{poly}(\kappa, \log(T), \log(1/\varepsilon))$.
- ***Round complexity:*** $r = \mathsf{polylog}(T)$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time} = (|x| + S) \cdot \mathsf{poly}(\kappa, \log(T), \log(1/\varepsilon))$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

Proof of Theorem 4.22. The argument system is derived from the honing protocol of Theorem 4.21. Let $\mathcal{M}$ be the Turing machine describing the computation, with initial state u_{start} and accepting state u_{accept}, and let x be the input.

The protocol. The protocol proceeds in iterations $j \leftarrow 1, \ldots r = O(\log T)$. In the j-th iteration, the prover and the verifier begin with the description of a computation whose total size decreases as the iterations proceed. The computation for the j-th iteration is described by a table $U^{(j)}$ of configurations. This table is explicitly sent by prover to the verifier. In the beginning of the j-th iteration the prover and verifier also agree on a computation length $\ell^{(j)}$ and a predicate $\phi^{(j)}$ over the concatenated tableaux $\mathcal{T}_{U^{(j)},x,S,\ell^{(j)}}$. The predicate is $\rho^{(j)}$-succinct, with size bounded by $\mathsf{poly}(|U^{(j)}|, \mathit{length}^{(j)})$ and depth bounded by $\mathsf{poly}(\log T)$. As the iterations proceed, the computation length will shrink but the size of the table will not grow. After the final iteration, the table size and computation length will both be $\mathsf{polylog}(T)$, and the verifier can check on its own whether the final predicate is satisfied by the tableaux in $\mathsf{polylog}(T)$ time.

Input for first iteration. In the first iteration, the table $U^{(1)}$ includes $k = \mathsf{polylog}(T)$ configurations: the initial configuration of the machine, and the k intermediate configurations (each proceeding the previous one by (T/k) computation steps). The computation length is $\ell^{(1)} = (T/k)$ and the predicate $\phi^{(1)}$ checks that the concatenated tableau describes an accepting computation of the machine $\mathcal{M}$ on the input x: i.e., that the first configuration is the initial configuration, that the $(k-1)$ proceeding configurations are the appropriate mid-points, and that the computation reaches the accepting state.

Iteration j. In the j-th iteration, the prover and the verifier run the honing protocol of Theorem 4.21 on the table $U^{(j)}$ with computation length $\ell^{(j)}$, the predicate $\phi^{(j)}$ and soundness error $1/\mathsf{polylog}(T)$. There is only one set $A^{(j)} = [U^{(j)}]$, which includes all the rows in the table. The honing protocol outputs a subset $B^{(j)} \subseteq [U^{(j)}]$ of the configurations and an LDE predicate $\psi^{(j)}$ over their concatenated tableax.

Assume that $|B^{(j)}| \leq |U^{(j)}|/2$, i.e. that at most half of the rows survive (we expect only $O(\log\log(T)/\log(T))$ of the rows to survive). For each surviving row of the table, which induces a tableau of length $\ell^{(j)}$, the prover also sends the mid-point of its computation. This gives a new table of size $2|B^{(j)}| \leq |U^{(j)}|$, which is explicitly known to the prover and the verifier. This table

will be used as $U^{(j+1)}$ for the next iteration. The new computation length is $\ell^{(j+1)} = \ell^{(j)}/2$. The new predicate $\phi^{(j+1)}$ checks that:

1. the new configurations are indeed the mid-points of their respective computations.
2. the output predicate $\psi^{(j)}$ from the previous iteration accepts the concatenated tableaux.
3. the initial states *equal* $U^{(j+1)}$: for each $a \in [|U^{(j+1)}|]$ the first state in the a-th tableau given as input to $\psi^{(j)}$ equals $U^{(j+1)}[a]$.

Finally, if $|B^{(j)}| > |U^{(j)}|/2$, then the prover and the verifier "ignore" the results of the honing protocol and use the same table, computation length, and predicate for the next iteration. Since the expected fractional size of $B^{(j)}$ is $\widetilde{\Theta}(1/\log(|U^{(j)}| \cdot \ell^{(j)}))$, by Markov's inequality this should happen with probability smaller than say 0.01 in each iteration (the probability is only over the verifier's coin tosses). If it happens in more than half of the $O(\log(T))$ iterations in the protocol, then the verifier aborts and accepts.

Completeness and soundness. Completeness follows by design. For soundness, first observe that the probability that there are too many "ignored" iterations is smaller than $1/\mathsf{poly}(T)$ by the analysis above. Otherwise, in the final iteration the computation length is indeed $\mathsf{polylog}(T)$ and the verifier can check for itself whether the predicate accepts.

If the prover's initial claim in the argument system is false, then the correct tableau leads to a rejecting state, and the there is no table that would make $\phi^{(1)}$ accept, so $U^{(1)}$ is at infinite distance from an input whose induced tableaux would make $\phi^{(1)}$ accept.

In the j-th iteration, if the input is far from (a table that induces tableaux) making the predicate accept, then by the soundness of the honing protocol, the output predicate should also reject the tableaux induced by the subset of rows in $B^{(j)}$ with all but $1/\mathsf{polylog}(T)$ probability. This means that there is no table whose induced tableau would make $\phi^{(j+1)}$ accept (since that predicate checks that the table states are equal the subset of rows in $B^{(j)}$, and internal consistency of the induced tableaux). Thus the input table for the next iteration is far from (a table that induced tableaux) making the next's iteration's predicate accept.

Taking a Union bound over all the iterations, w.h.p. the final iteration results in a predicate that rejects the tableaux induced by its table, and the verifier (who can check this on its own) will also reject. The soundness error can be reduced by parallel repetition [BIN97, Hai13].

Complexity. The circuits describing the subset for reach iteration are fully succinct (the subset is the complete set). The predicates are $(S \cdot \mathsf{polylog}(T))$-succinct, because each predicate (except the first) includes the entire alleged table of states. Thus, in each call to the honing protocol of Theorem 4.21, the communication complexity is $(S \cdot \mathsf{poly}(\kappa, \log(T))$, there are $\mathsf{polylog}(T)$ rounds, the verifier's runtime is $(|X| + S) \cdot \mathsf{poly}(\kappa, \log(T))$ and the honest prover runs in $\mathsf{poly}(T)$ time. Note that the verifier can compute the set of surviving rows in $\mathsf{polylog}(T)$ time (and each time the set of rows for the next iteration is the full set in the new table). The complexity bounds for the theorem follow. □

5 Improved Unconditional DEIP

We detail our improved *unconditionally sound* DEIP for bounded-space computations, following the overview in Section 2.2. The construction follows from an unconditionally sound recursive honing protocol (by the same protocol used in Section 4.4 for argument systems). The main difference from the construction of Section 4 is that the resulting protocol is less efficient. This is because the recursive construction makes *two* recursive calls in each level of the recursion. On the other hand, soundness is unconditional and uses no cryptographic assumptions or machinery.

The construction in this section is also simpler in that it suffices to construct honing protocols for the single-set case (i.e. there is no collection of nested subsets), see Remark 3.15. We restrict our attention to this case throughout.

Organization The base of the recursion is in Section 5.1. The recursive step and its analysis are in Section 5.2. We put things together to obtain the resulting honing protocol in Section 5.3, and we derive the unconditional DEIP in Section 5.4.

5.1 Base of The Recursion

We use an unconditionally sound IPP at the base of the recursion: the IPP from [RR20], see Theorem 3.28. That construction gives the following parameters:

Corollary 5.1. *There is an* unconditionally sound *honing protocol for the (single set) multi-tableau language as follows: for an instance $(\mathcal{M}, x, 1^S, \ell^{\mathsf{base}}, U, \phi)$ and (absolute) row distance $\delta^{\mathsf{base}}_{ABS} \leq |U|$, take $T = |U| \cdot \ell^{\mathsf{base}}$. We require that the predicate ϕ is ρ^{base}-succinct with depth D and size $\mathsf{poly}(T)$, and that $|x|, S, \rho^{\mathsf{base}} \leq T$. The protocol is public-coins with perfect completeness. For desired soundness ε, it has:*

- ***Honing parameter:*** $\eta^{\mathsf{base}} = O\left(\frac{\log(1/\varepsilon)}{\delta^{\mathsf{base}}_{ABS}}\right)$.
- ***Communication complexity:*** $\mathsf{cc}^{\mathsf{base}} = (\delta^{\mathsf{base}}_{ABS} \cdot S + \ell^{\mathsf{base}} + D) \cdot \mathsf{polylog}(T)$.
- ***Round complexity:*** $r^{\mathsf{base}} = (\ell^{\mathsf{base}} + D) \cdot \mathsf{polylog}(T)$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time}^{\mathsf{base}} = (|x| + \rho^{\mathsf{base}} + \delta^{\mathsf{base}}_{ABS} \cdot S + \ell^{\mathsf{base}} + D) \cdot \mathsf{polylog}(T)$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time}^{\mathsf{base}} = \mathsf{poly}(T)$.

The output set B and predicate ψ are $\mathsf{polylog}(T)$-succinct, and computable by circuits of depth $\mathsf{polylog}(T)$ and size $|U| \cdot S \cdot \mathsf{polylog}(T)$ (for B) and $|B| \cdot S \cdot \mathsf{polylog}(T)$ (for ψ). We emphasize that the succinctness and complexities of these circuits are independent of those of the input predicate ϕ).

Proof. The honing protocol follows directly from the IPP of Theorem 3.28, applied to the circuit that takes as input the table U, computes (in parallel) the length-ℓ^{base} tableau that starts in each state of U, and then applies the predicate ϕ. See also the proof of Corollary 4.1 for further details. □

5.2 The Recursive Step

We formalize the recursive step of the unconditional construction. The main result of this section shows how to construct an unconditional honing protocol for computations of length $(k \cdot \ell)$ using *two* oracle calls to a protocol for computations of length ℓ. The construction is in Figure 2. We prove its soundness and honing properties in Lemma 5.3 and analyze its complexity in Lemma 5.2.

Lemma 5.2 (Unconditional recursive construction complexity)**.** *For every input tuple $(\mathcal{M}, x, 1^S, \ell^{\mathsf{outer}}, U)$ and ρ^{outer}-succinct predicate ϕ^{outer}, for parameters d and $k \leq \ell^{\mathsf{outer}}$, let $T = |U| \cdot \ell^{\mathsf{outer}}$. We assume that $T \geq |x|, S, d, \rho^{\mathsf{outer}}$ and that the predicate ϕ^{outer} has size $S^{\mathsf{outer}} \leq \mathsf{poly}(T)$ and depth D^{outer}.*

Suppose that the following conditions hold for the inner honing protocol: let $\mathsf{cc}^{\mathsf{inner}}$, r^{inner}, $\mathcal{V}\mathsf{time}^{\mathsf{inner}}$, $\mathcal{P}\mathsf{time}^{\mathsf{inner}}$ bound the communication, rounds, verifier runtime and prover runtime (respectively) of the inner honing protocol when it is run as follows:

- *the parameters are η^{inner}, $\delta^{\mathsf{inner}}_{ABS}$, $|x|$, S, ℓ^{outer}/k,*
- *the table size satisfies $|U^{\mathsf{inner}}| \leq \max\{k \cdot |U|, |U| \cdot (1 + \eta^{\mathsf{inner}} \cdot k^2)\}$.*
- *the predicate ϕ^{inner} is ρ^{inner}-succinct, has size at most S^{inner} and depth at most D^{inner} (the succinctness and complexity of ϕ^{inner} can depend on those of ϕ^{outer}).*

  Suppose that the subset of surviving rows B^{inner} and the output predicate ψ^{inner} are both computable by μ^{inner}-succinct circuits of size $S^{\mathsf{inner-out}}$ and depth $D^{\mathsf{inner-out}}$.

  We require that $\rho^{\mathsf{inner}} \geq \max\{\rho^{\mathsf{outer}}, 2\mu^{\mathsf{inner}} + d \cdot S \cdot \mathsf{polylog}(T)\}$, that $S^{\mathsf{inner}} \geq \max\{S^{\mathsf{outer}}, 2S^{\mathsf{inner-out}}\} + \mathsf{poly}(T)$, and that $D^{\mathsf{inner}} \geq \max\{D^{\mathsf{outer}}, 2D^{\mathsf{inner-out}}\} + \mathsf{polylog}(T)$.

Under the above conditions, the complexities of the outer honing protocol are:

- *communication complexity $\mathsf{cc}^{\mathsf{outer}} = 2 \cdot \mathsf{cc}^{\mathsf{inner}} + S \cdot d \cdot \mathsf{polylog}(T)$.*
- *round complexity $r^{\mathsf{outer}} = 2 \cdot r^{\mathsf{inner}} + 1$.*
- *prover runtime $\mathcal{P}\mathsf{time}^{\mathsf{outer}} = 2 \cdot \mathcal{P}\mathsf{time}^{\mathsf{inner}} + \mathsf{poly}(T)$.*
- *verifier runtime $\mathcal{V}\mathsf{time}^{\mathsf{outer}} = 2 \cdot \mathcal{V}\mathsf{time}^{\mathsf{inner}} + S \cdot d \cdot \mathsf{polylog}(T)$.*
- *the set B^{outer} of surviving indices is $2\mu^{\mathsf{inner}}$-succinct and computable by a circuit of size $2S^{\mathsf{inner-out}} + \mathsf{poly}(T)$ and depth $D^{\mathsf{inner-out}} + \mathsf{polylog}(T)$.*
- *the predicate ψ^{outer} is $3\mu^{\mathsf{inner}}$-succinct and computable by a circuit of size $3S^{\mathsf{inner-out}} + \mathsf{poly}(T)$ and depth $2D^{\mathsf{inner-out}} + \mathsf{polylog}(T)$.*

Proof. We analyze each component of the outer protocol's complexity.

Inner protocol calls. The first call operates on table U^{expd} of size $k \cdot |U|$ with the predicate ϕ^{inner}, which is $\rho^{\mathsf{outer}} \leq \rho^{\mathsf{inner}}$-succinct and has size $S^{\mathsf{outer}} + \mathsf{poly}(T)$ and depth $D^{\mathsf{outer}} + \mathsf{polylog}(T)$. By the lemma conditions, this uses $\mathsf{cc}^{\mathsf{inner}}$ communication, r^{inner} rounds, $\mathcal{V}\mathsf{time}^{\mathsf{inner}}$ verifier time, and $\mathcal{P}\mathsf{time}^{\mathsf{inner}}$ prover time, producing output set B' and predicate ψ' with the stated succinctness, size, and depth bounds.

Outer protocol Π^{outer}, with two recursive calls to inner protocol Π^{inner}

Parameters: honing η^{inner}, absolute proximity $\delta^{\mathsf{inner}}_{\mathsf{ABS}}$, expansion k, checksum d.
Input: $(\mathcal{M}, x, 1^S, \ell^{\mathsf{outer}} = (k \cdot \ell), U, \phi^{\mathsf{outer}})$. Take $T = |U| \cdot \ell^{\mathsf{outer}}$.

1. **Checksum:** The prover sends a d-checksum $\widetilde{C_U}$ for the table of states U.
2. **First recursive call:** Let U^{expd} be the k-fold expansion of U (Definition 3.16), containing the k mid-point states of the $(k \cdot \ell)$-length computation induced by each state in U. Let ϕ^{inner} be a predicate on the length-ℓ tableaux of U^{expd} that checks:
   (a) internal consistency: the tableaux of each k-row block of U^{expd} correspond to a longer computation of length $(k \cdot \ell)$.
   (b) ϕ^{outer} accepts the concatenation of the length-ℓ tableaux of U^{expd}.

   The prover and verifier run Π^{inner} on $(\mathcal{M}, x, 1^S, \ell, U^{\mathsf{expd}}, \phi^{\mathsf{inner}})$ with parameters $\eta^{\mathsf{inner}}, \delta^{\mathsf{inner}}_{\mathsf{ABS}}$. Let B' be the output set and ψ' be the output predicate. Let $B \subseteq [|U|]$ be the k-fold index contraction of B', and let ψ be the corresponding predicate over the $(k \cdot \ell)$-length tableaux of $U|_B$.
3. **Second recursive call:** Define the table U'' that includes each state in U, and for the states in B also includes the block of k intermediate states of that state's $(k \cdot \ell)$-length computation. Define the predicate ϕ'' on the length-ℓ tableaux of U'' that checks:
   (a) the first states in the tableaux of the states that are in U have checksum $\widetilde{C_U}$.
   (b) the blocks corresponding to states in B are internally consistent and their concatenated tableaux satisfy ψ.

   The prover and verifier run Π^{inner} on $(\mathcal{M}, x, 1^S, \ell, U'', \phi'')$ with parameters $\eta^{\mathsf{inner}}, \delta^{\mathsf{inner}}_{\mathsf{ABS}}$. Let $B'' \subseteq [|U''|]$ be the output set and (ψ'') the output predicates.
4. **Output:** The output set $B^{\mathsf{outer}} \subseteq [|U|]$ includes: (a) the rows of U that were not in B but survived in B'', we call these rows B^{outer}_1, and (b) the rows of U in B for which at least one of their k mid-points survived in B'', we call these rows B^{outer}_2.

   The output predicate ψ^{outer} over the $(k \cdot \ell)$-length tableaux of the rows in B checks that the predicate ψ'' is satisfied by the (concatenation of) the sequence of length-ℓ tableau that includes: (a) the first ℓ configurations for each row in B^{outer}_1, and (b) for each row in B^{outer}_2, the length-ℓ block (or blocks) corresponding to the midpoint (or midpoints) that survived in B''.

Figure 2: Unconditional Recursive Outer Protocol Π^{outer}

The second call operates on table U'' of size at most $|U| \cdot (1 + k^2 \cdot \eta^{\mathsf{inner}})$. The predicate ϕ'' includes: (i) the checksum $\widetilde{C_U}$ (of description size $S \cdot d \cdot \mathsf{polylog}(T)$), and (ii) a description of the set B of surviving rows from the first call, and (iii) the predicate ψ' from the first call to check the intermediate tableaux of rows in B. Thus ϕ'' is $(2\mu^{\mathsf{inner}} + S \cdot d \cdot \mathsf{polylog}(T))$-succinct. It has size at most $2S^{\mathsf{inner-out}} + \mathsf{poly}(T)$ and depth at most $2D^{\mathsf{inner-out}} + \mathsf{polylog}(T)$. By the Lemma conditions, the second call also uses $\mathsf{cc}^{\mathsf{inner}}$, r^{inner}, $\mathcal{V}\mathsf{time}^{\mathsf{inner}}$, $\mathcal{P}\mathsf{time}^{\mathsf{inner}}$ and produces B'', ψ'' with the stated succinctness, size and depth bounds.

Checksum overhead. In Step 1, the prover sends a d-checksum for U, requiring $S \cdot d \cdot \mathsf{polylog}(T)$ communication. This adds 1 round and the verifier needs to read the checksum, which adds to its runtime.

Output set B^{outer}. The set B^{outer} is constructed from B' and B'' as follows (see Step 4):

- Compute B (the k-fold contraction of B'), which indicates which rows of U have at least one surviving mid-point from the first call.
- For each row $i \in [|U|]$: (a) if $i \notin B$ and $i \in B''$, include i in B_1^{outer}; (b) if $i \in B$ and at least one of its k mid-points survived in B'', include i in B_2^{outer}.

The circuit for B^{outer} is thus $2\mu^{\mathsf{inner}}$-succinct and can be implemented with size $\mathsf{size}(B') + \mathsf{size}(B'') + \mathsf{poly}(|U|, k, \log T)$ by composing the circuits for B' and B'' with additional $\mathsf{poly}(|U|, k, \log T)$ logic for the contraction and combination.

For the depth: the contraction of B' and the combination logic add only $\mathsf{polylog}(T)$ additional depth (they involve $\mathsf{polylog}(T)$-depth arithmetic for index computations and comparisons). The same is true for the contraction of B''. B' and B'' can be evaluated independently and their results combined in $\mathsf{polylog}(T)$-depth. Thus, the total depth is $\max\{\mathsf{depth}(B'), \mathsf{depth}(B'')\} + \mathsf{polylog}(T)$.

Output predicate ψ^{outer}. The predicate ψ^{outer} takes as input the $(k \cdot \ell)$-tableaux for the row in B^{outer} and applies the predicate ψ'' to (the concatenation of) the following ℓ-length tableaux:

- For rows in B_1^{outer}: the first ℓ steps.
- For rows in B_2^{outer}: the block(s) of ℓ steps corresponding to surviving mid-point(s) in B''.

The circuit for ψ^{outer} needs to: (i) determine which case applies (using membership in B' and B''), (ii) extract the appropriate ℓ-tableau segment, and (iii) apply ψ''. This requires the circuit for ψ'' plus the circuits for B', B'' (to determine membership), plus $\mathsf{poly}(|U|, k, \log T)$ additional logic for index selection. Thus $\mathsf{size}(\psi^{\mathsf{outer}}) \leq \mathsf{size}(\psi'') + \mathsf{size}(B') + \mathsf{size}(B'') + \mathsf{poly}(|U|, k, \log T)$.

For the depth: the index selection logic adds $\mathsf{polylog}(T)$ depth. The sets B' and B'' can be computed in parallel, but the input to ϕ'' depends on their output. Thus, the total depth is $\mathsf{depth}(\psi'') + \max\{\mathsf{depth}(B'), \mathsf{depth}(B'')\} + \mathsf{polylog}(T)$. □

Lemma 5.3 (Unconditional recursive construction soundness)**.** *Suppose that Π^{inner} is a $(\eta^{\mathsf{inner}}, \delta_{\mathsf{ABS}}^{\mathsf{inner}}, \ell)$-honing protocol with soundness error ε_s (see Definition 3.35), where $\eta^{\mathsf{inner}} \leq 1/k^3$ and $\delta_{\mathsf{ABS}}^{\mathsf{inner}} \leq d$.*

Then Π^{outer} is a $(\eta^{\mathsf{outer}}, \delta_{\mathsf{ABS}}^{\mathsf{outer}}, k \cdot \ell)$-honing protocol, where:

- $\eta^{\mathsf{outer}} \leq \left(1 + \frac{1}{k}\right) \cdot \eta^{\mathsf{inner}}$.

- $\delta^{\mathsf{outer}}_{ABS} = \frac{\delta^{\mathsf{inner}}_{ABS}}{k} + d$.
- *The soundness error is at most* $2\varepsilon_s$.

Proof of Lemma 5.3. We begin by bounding the honing parameter. The proof of soundness follows.

Honing parameter. Following the construction in Figure 2, the output set B^{outer} is formed from two sources (see Step 4):

- Rows of U that were not in B (the contraction of the set output by the first recursive call), but survived in B'' (the set output by the second recursive call). These rows are not expanded in the input U'' to the second recursive call, thus the expected number of surviving rows of this type is at most $|U| \cdot \eta^{\mathsf{inner}}$.
- Rows of U that survived in B, for which at least one of their k mid-points survived in B''. By the honing guarantee of the first call, the expected *fraction* of rows of U in B is at most $k \cdot \eta^{\mathsf{inner}}$. Each row in B is expanded to k midpoint-rows in the input U'' to the second recursive call. By linearity of expectation, the expected number of surviving rows of this type is at most $|U| \cdot (k \cdot \eta^{\mathsf{inner}})^2$.

Putting this together, the expected total fraction of surviving rows is:

$$\eta^{\mathsf{inner}} + \left(k \cdot \eta^{\mathsf{inner}}\right)^2 \leq \left(1 + \frac{1}{k}\right) \cdot \eta^{\mathsf{inner}},$$

where the inequality holds because $\eta^{\mathsf{inner}} \leq 1/k^3$ implies $(k \cdot \eta^{\mathsf{inner}})^2 \leq \eta^{\mathsf{inner}}/k$.

Bounding the soundness error. Suppose the table U is $\delta^{\mathsf{outer}}_{\mathsf{ABS}}$-far from a table whose tableaux satisfy the input predicate ϕ. Let $\widetilde{U}$ be the matrix closest in row-distance to U that has checksum $\widetilde{C_U}$. By the properties of the checksum, there is at most one such matrix at distance d.

First recursive call (Step 2). Suppose $\widetilde{U}$ is at distance at most d from U (the second call handles the case where this is not true). By a triangle inequality, $\widetilde{U}$ is $(\delta^{\mathsf{outer}}_{\mathsf{ABS}} - d)$-far from any table whose tableaux satisfy ϕ. We conclude that $\widetilde{U}^{\mathsf{expd}}$, the k-fold expansion of $\widetilde{U}$, is at least $k \cdot (\delta^{\mathsf{outer}}_{\mathsf{ABS}} - d) \geq \delta^{\mathsf{inner}}_{\mathsf{ABS}}$-far from satisfying ϕ^{inner}: see the analogous proof of Claim 4.19.[10]

By the soundness of Π^{inner}, with probability at least $1 - \varepsilon_s$, either the recursive call rejects, or the output predicate ψ' does not accept the tableaux of $(\widetilde{U}^{\mathsf{expd}})|_{B'}$. In the latter case, the derived predicate ψ does not accept the tableaux of $\widetilde{U}|_B$. We assume for the remainder of the proof that this latter case holds.

[10] In more detail: let Z be a table whose length-ℓ tableaux are accepted by ϕ^{inner}. Since ϕ^{inner} also checks internal consistency, it must be the case that $Z = (Z^{\mathsf{ctrt}})^{\mathsf{expd}}$. Since ϕ^{inner} checks that ϕ^{outer} accepts the concatenation of the length-ℓ tableaux of each k-block, for any Z whose k-fold contraction Z^{ctrt} is at absolute row-distance less than $(\delta^{\mathsf{outer}}_{\mathsf{ABS}} - d)$ from $\widetilde{U}$, it must be the case that ϕ^{outer} rejects Z^{expd}. By Fact 3.12 and Remark 3.13 we conclude that $Z = (Z^{\mathsf{ctrt}})^{\mathsf{expd}}$ is at least $k \cdot (\delta^{\mathsf{outer}}_{\mathsf{ABS}} - d)$ far from $\widetilde{U}^{\mathsf{expd}}$ (i.e. the distance of the k-fold expansion blows up by a multiplicative factor of k).

Ensuring distance. We show that U is d-far from any table V that both: (i) has checksum $\widetilde{C_U}$, and (ii) the $(k \cdot \ell)$-tableaux of $V|_B$ satisfy ψ. If $\widetilde{U}$ (the closest table to U with checksum $\widetilde{C_U}$) is itself d-far from U then this follows immediately. Otherwise, assume for contradiction that there exists V that is d-close to U and satisfies the conditions (i) and (ii) above. By the uniqueness of $\widetilde{U}$, we must have $V = \widetilde{U}$. But we know from the soundness of the first call that ψ does not accept the tableaux of $\widetilde{U}|_B$.

Second recursive call (Step 3). Consider the expanded table U'' constructed in Step 3. By the above, U'' is at absolute distance at least d from any table V'' whose ℓ-tableaux satisfy ϕ''.[11] By the lemma's condition, $\delta^{\mathsf{inner}}_{\mathsf{ABS}} \leq d$, so U'' is at least $\delta^{\mathsf{inner}}_{\mathsf{ABS}}$-far from satisfying ϕ''.

By the soundness of Π^{inner}, with probability at least $1 - \varepsilon_s$, either the second recursive call rejects, or the output predicate ψ'' does not accept the tableaux of $U''|_{B''}$. In the latter case, the derived predicate ψ^{outer} does not accept the tableaux of B^{outer}.

Final bound. By a union bound, the total soundness error is at most $2\varepsilon_s$. □

5.3 The Resulting Honing Protocol

Theorem 5.4. *There is an* unconditionally sound *honing protocol for the (single-set) multi-tableau language as follows: for an instance $(\mathcal{M}, x, 1^S, \ell, U, \phi)$, take $T = 2|U| \cdot \ell$, where we require that $|x|, S, \rho \leq T$, and that the predicate ϕ is ρ-succinct with $\mathsf{poly}(T)$ size and $2^{O(\sqrt{\log T})}$ depth. The protocol is public-coin with perfect completeness. For desired soundness $\varepsilon \geq 1/T$, it has:*

- ***Absolute distance:*** $\delta_{ABS} = \min\{\widetilde{O}(2^{3\sqrt{\log T}}), |U|\}$.
- ***Honing parameter:*** $\eta = \widetilde{O}\left(\frac{\log(1/\varepsilon)}{\delta_{ABS}}\right)$.
- ***Communication complexity:*** $\mathsf{cc} = S \cdot 2^{O(\sqrt{\log T})}$.
- ***Round complexity:*** $r = 2^{O(\sqrt{\log T})}$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time} = (|x| + S + \rho) \cdot 2^{O(\sqrt{\log T})}$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

The output set B and predicate ψ are $2^{O(\sqrt{\log T})}$-succinct and computable by circuits of size $\mathsf{poly}(T)$ and depth $2^{O(\sqrt{\log T})}$ (these are independent of the succinctness, size and depth of the input predicate ϕ).

Proof. The proof uses the same recursive structure as Theorem 4.21, with the protocol of Corollary 5.1 at the base and the recursive step of Figure 2 (Lemmas 5.2 and 5.3). We assume w.l.o.g. that $|U| \geq \delta_{\mathsf{ABS}}$ (otherwise we pad the input with dummy states until the bound is satisfied).

[11] In more detail: any V'' satisfying ϕ'' must have its rows corresponding to U agree with some table that has checksum $\widetilde{C_U}$ (since ϕ'' checks this in Step 3(a)), and the length-ℓ tableaux of its k-row blocks corresponding to B (one block for each state in B) must satisfy ψ (and each block must correspond to a single length-$(k \cdot \ell)$ computation, see Step 3(b)). By the distance-ensuring argument above, we conclude any such table V'' must be at row-distance greater than d from U''. Note that we do not have a k-fold distance blowup here (because the rows outside of B are not expanded).

We number the layers of the recursion from 0 (the top / outermost layer) to r (the bottom / base). For each level $j \in [1, r]$, we construct the protocol for level $j-1$ using the recursive construction of Figure 2, with the protocol for level j as the inner protocol. At the base of the recursion we use the unconditional IPP of Corollary 5.1.

Bounding computation sizes. At the top level of the recursion $\ell^{(0)} = \ell$ is the length of each tableau, $u^{(0)} = |U|$ is the input table size and $T^{(0)} = u^{(0)} \cdot \ell^{(0)}$ is the total size of the computation.

The j-th level of the recursion gives protocols for tableaux of length $\ell^{(j)} = \ell/k^j$. We take $u^{(j)}$ to be a bound on the table sizes for calls to this protocol. If we unwind the recursion, there are 2^j executions of the protocol constructed in the j-th level: each execution of the protocol of level $j-1$ on a table $U^{(j-1)}$ results in two executions of the protocol of level j: the first execution is on the table $U^{(j-1)^{\mathsf{expd}}}$, which has size at most $k \cdot u^{(j-1)}$. The second call, on the table $U^{(j-1)''}$ has size at most $k \cdot (1 + \eta k^2) \cdot u^{(j-1)} \leq k \cdot (1 + 1/k) \cdot u^{(j-1)}$ (we use the fact that $\eta < 1/k^2$). We take $T^{(j)} = u^{(j)} \cdot \ell^{(j)}$ to be a bound on the total size of the computation in calls to the protocol in the j-th level of the recursion. We have:

$$\forall j \in [0, r] : u^{(j)} = k^j \cdot \left(1 + \frac{1}{k}\right)^j \cdot u^{(0)},$$
$$T^{(j)} = u^{(j)} \cdot \ell^{(j)} = \left(1 + \frac{1}{k}\right)^j \cdot T^{(0)}$$

Looking ahead, the recursion depth will be (much) smaller than $k/2$, so the computation size at the base of the recursion can be upper bound by T:

$$\forall j \in [0, r] : T^{(0)} \leq T^{(j)} \leq T \triangleq 2u^{(0)} \cdot \ell^{(0)}$$

We fix $k = 2^{\sqrt{\log T}}$ (the same k at all levels). The j-th level gives protocols for tableaux of length $\ell^{(j)} = \ell/k^j$. We stop the recursion at depth r when $\ell^{(r)} = k$, i.e.:

$$r = \left\lceil \log_k \frac{\ell^{(0)}}{k} \right\rceil = O\left(\sqrt{\log T}\right).$$

Note that indeed $r \leq k/2$, as assumed above (because $r = O(\sqrt{\log T}) \ll k = 2^{\sqrt{\log T}}$).

Parameter setting. For $j \in [0, r]$:

- **Checksum parameter.** $d^{(j)} = \lceil (1+1/k) \cdot d^{(j+1)} \rceil$, where $d^{(r)} = k^3 \cdot \mathsf{polylog}(T) = \widetilde{O}(2^{3\sqrt{\log T}})$ (the definition of $d^{(r)}$ is syntactic, the protocol at the base of the recursion does not use the recursive construction and has no additional checksum).
- **Absolute distance.** $\delta^{(j)}_{\mathsf{ABS}} = (1 + 1/k) \cdot d^{(j)}$.
- **Soundness error.** $\varepsilon^{(j)} = 2\varepsilon^{(j+1)}$, where $\varepsilon^{(r)} = 1/2^{r+2}$.
- **Honing parameter.** $\eta^{(j)} = (1 + 1/k) \cdot \eta^{(j+1)}$, where $\eta^{(r)} = O(\log(1/\varepsilon^{(r)})/\delta^{(r)}_{\mathsf{ABS}})$.

The conditions of Lemma 5.3 at each level require $\eta^{(j+1)} \leq 1/k^3$ and $\delta_{\mathsf{ABS}}^{(j+1)} \leq d^{(j)}$. The latter holds by construction. For the former, $\forall j \in [1, r]$:

$$\eta^{(j)} \leq \left(1 + \frac{1}{k}\right)^r \cdot \eta^{(r)} \leq 2\eta^{(r)} = \frac{O\left(\log(1/\varepsilon^{(r)})\right)}{\mathsf{polylog}(T) \cdot k^3} \leq \frac{1}{k^3},$$

where the last inequality holds for T large enough that the $\mathsf{polylog}(T)$ term is larger than the $O\left(\log(1/\varepsilon^{(r)})\right) = O(r) = o(\log(T))$ term.

Soundness. At the base of the recursion, the IPP of Corollary 5.1 achieves soundness error $\varepsilon^{(r)}$. By Lemma 5.3, each level of the recursion at most doubles the soundness error (union bound over two recursive calls). Thus:

$$\varepsilon^{(0)} \leq 2^r \cdot \varepsilon^{(r)}.$$

We set $\varepsilon^{(r)} = 1/2^{r+2}$, so that $\varepsilon^{(0)} \leq 1/4$. We amplify the soundness error from $1/4$ to ε using parallel repetition with $O(\log(1/\varepsilon))$ overhead.

Since $r \leq k/2$, we have $d^{(j)}, \delta_{\mathsf{ABS}}^{(j)} = \widetilde{O}(k^3) = \widetilde{O}(2^{3\sqrt{\log T}})$ at all levels.

Circuit succinctness, size and depth. By construction (and by the bound on the size of the initial input predicate ϕ), the input predicates to all calls have size $\mathsf{poly}(T)$. We show by induction that the output sets and predicates at level $j \in [0, r]$ are $(2^{O(r-j)} \cdot \mathsf{polylog}(T))$-succinct, with size $\mathsf{poly}(T)$ and depth $(2^{O(r-j)} \cdot \mathsf{polylog}(T))$. At the base of the recursion, Corollary 5.1 implies that the circuits have succinctness and depth $\mathsf{polylog}(T)$ and size $u^{(r)} \cdot \mathsf{polylog}(T)$. By Lemma 5.2, the succinctness, depths and sizes grow by a constant multiplicative factor. The size grows by an additional fixed additive $\mathsf{poly}(T)$ term and the depth grows by an additive $\mathsf{polylog}(T)$ term.

We also use induction to bound the succinctness, size and depth of the input predicates in the calls to the protocols at each level of the recursion (so we can apply the complexity bounds of Lemma 5.2). Let $\rho^{(j)}$ and $D^{(j)}$ denote the succinctness and depth of the input predicate at level j. We claim that for every $j \in [0, r]$:

$$\rho^{(j)} \leq \max\{\rho, S \cdot 2^{O(\sqrt{\log T})}\}, \qquad D^{(j)} \leq 2^{O(\sqrt{\log T})},$$

and the size is always $\mathsf{poly}(T)$. At level 0 this follows from the conditions on ϕ in the theorem statement. For the inductive step, consider the two recursive calls from level j to level $j+1$:

- The first call's input predicate is just $\phi^{(j)}$, so it is $\rho^{(j)}$-succinct, with size $\mathsf{size}(\phi^{(j)}) + \mathsf{poly}(T)$ and depth $D^{(j)} + \mathsf{polylog}(T)$.
- By the output-set/output-predicate bounds above, the second call's input predicate is built from circuits of succinctness and depth $2^{O(r-j)} \cdot \mathsf{polylog}(T) \leq 2^{O(\sqrt{\log T})}$ and size $\mathsf{poly}(T)$, plus the checksum of succinctness $S \cdot d^{(j)} \cdot \mathsf{polylog}(T) = S \cdot 2^{O(\sqrt{\log T})}$. Thus it is also within the claimed bounds.

Note that these bounds on the input and output predicate complexities and succinctness satisfy the conditions in Lemma 5.2.

The output set B and predicate ψ at level 0 are therefore $2^{O(\sqrt{\log T})}$-succinct, have size $\mathsf{poly}(T)$, and have depth $2^{O(\sqrt{\log T})}$, as stated in the theorem.

Protocol complexities: base of the recursion. At the base level r, with $\delta_{\mathsf{ABS}}^{(r)} = 2^{O(\sqrt{\log T})}$, $\ell^{(r)} = k = 2^{\sqrt{\log T}}$, depth $D^{(r)} = 2^{O(\sqrt{\log T})}$, and soundness error $\varepsilon^{(r)} = 2^{-O(\sqrt{\log T})}$, the complexities from Corollary 5.1 are:

- $\mathsf{cc}^{(r)} = (\delta_{\mathsf{ABS}}^{(r)} \cdot S + \ell^{(r)} + D^{(r)}) \cdot \mathsf{polylog}(T) = S \cdot 2^{O(\sqrt{\log T})}$.
- $r^{(r)} = \ell^{(r)} \cdot \mathsf{polylog}(T) = 2^{O(\sqrt{\log T})}$.
- $\mathcal{V}\mathsf{time}^{(r)} = (|x| + \rho^{(r)} + \delta_{\mathsf{ABS}}^{(r)} \cdot S + \ell^{(r)} + D^{(r)}) \cdot \mathsf{polylog}(T) = (|x| + S + \rho) \cdot 2^{O(\sqrt{\log T})}$.
- $\mathcal{P}\mathsf{time}^{(r)} = \mathsf{poly}(T)$.

Complexity overheads: inductive step. By Lemma 5.2, at each level $j \in [r]$ the outer protocol runs two inner calls and adds a checksum overhead of $S \cdot d^{(j)} \cdot \mathsf{polylog}(T) = S \cdot 2^{O(\sqrt{\log T})}$ communication and verifier time, plus 1 round and $\mathsf{poly}(T)$ prover time. The inner calls already account for the cost of evaluating the predicates, so the only additional verifier work is reading the checksum.

Total complexity. Since each level runs two inner calls, unrolling the recursion gives:

$$\mathsf{cc}^{(0)} = 2^r \cdot \mathsf{cc}^{(r)} + \sum_{j=0}^{r-1} 2^j \cdot S \cdot 2^{O(\sqrt{\log T})} = 2^r \cdot S \cdot 2^{O(\sqrt{\log T})},$$

and analogously:

$$\mathcal{V}\mathsf{time}^{(0)} = 2^r \cdot \mathcal{V}\mathsf{time}^{(r)} + \sum_{j=0}^{r-1} 2^j \cdot S \cdot 2^{O(\sqrt{\log T})} = 2^r \cdot (|x| + S + \rho) \cdot 2^{O(\sqrt{\log T})},$$
$$r^{(0)} = 2^r \cdot r^{(r)} + r = 2^r \cdot 2^{O(\sqrt{\log T})}.$$

With $2^r = 2^{O(\sqrt{\log T})}$, after $O(\log(1/\varepsilon))$ parallel repetitions, the final complexities are:

- **Total communication:** $\mathsf{cc} = S \cdot 2^{O(\sqrt{\log T})}$.
- **Total rounds:** $r = 2^{O(\sqrt{\log T})} \cdot \log(1/\varepsilon)$.
- **Total verifier time:** $\mathcal{V}\mathsf{time} = (|x| + S + \rho) \cdot 2^{O(\sqrt{\log T})} \cdot \log(1/\varepsilon)$.
- **Total prover time:** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

□

5.4 The Resulting DEIP

Theorem 5.5 (Unconditional DEIP for bounded space)**.** *There is an* unconditionally sound *doubly-efficient interactive proof for space-$S = S(n)$ time-$T = T(n) \geq n$ computations. The protocol is public-coin with perfect completeness. For desired soundness $\varepsilon \geq 1/T$ it has:*

- ***Communication complexity:*** $\mathsf{cc} = S \cdot 2^{O(\sqrt{\log T})}$.

- ***Round complexity:*** $r = 2^{O(\sqrt{\log T})}$.
- ***Verifier runtime:*** $\mathcal{V}\mathsf{time} = (|x| + S) \cdot 2^{O(\sqrt{\log T})}$.
- ***Honest prover runtime:*** $\mathcal{P}\mathsf{time} = \mathsf{poly}(T)$.

Proof. The DEIP is derived from the honing protocol of Theorem 5.4, mirroring the proof of Theorem 4.22. Let $\mathcal{M}$ be the Turing machine with initial state u_{start} and accepting state u_{accept}, and let x be the input.

The protocol. The protocol proceeds in iterations $j = 1, \ldots, r = O(\log T)$. In the j-th iteration, the prover and verifier begin with an explicit table $U^{(j)}$ of configurations, an agreed computation length $\ell^{(j)}$, and a predicate $\phi^{(j)}$ over the concatenated tableaux $\mathcal{T}_{U^{(j)},\ell^{(j)}}$. The predicate is $\rho^{(j)}$-succinct with size $\mathsf{poly}(|U^{(j)}|, \ell^{(j)})$ and depth $2^{O(\sqrt{\log T})}$. As the iterations proceed, the computation length is halved; the table size does not grow.

We remark that we could reduce the computation length by larger factors, corresponding to the $\eta = \widetilde{O}\left(1/2^{3\sqrt{\log T}}\right)$ honing parameter obtained in Theorem 5.4, but this would not substantially improve the parameters of the resulting DEIP.

Input for the first iteration. Take $k = \mathsf{polylog}(T)$. The table $U^{(1)}$ consists of k configurations: the initial configuration of $\mathcal{M}$ on x, plus $k-1$ intermediate configurations spaced T/k steps apart. The prover explicitly sends these mid-points to the verifier in its first message. The computation length is $\ell^{(1)} = T/k$. The predicate $\phi^{(1)}$ checks that the concatenated tableau describes an accepting computation: the first configuration is the initial configuration, the last reaches u_{accept}, and the intermediate configurations are the correct mid-points (these intermediate configurations are hardwired into the predicate). This predicate is $(k \cdot S)$-succinct, has size $\mathsf{poly}(T)$, and depth $\mathsf{polylog}(T)$.

Iteration j. In the j-th iteration, the prover and verifier run the honing protocol of Theorem 5.4 on the table $U^{(j)}$ with computation length $\ell^{(j)}$, predicate $\phi^{(j)}$, and soundness error $\varepsilon_h = 1/(2\log T)$. The honing protocol outputs a subset $B^{(j)} \subseteq [|U^{(j)}|]$ and an LDE predicate $\psi^{(j)}$ over the concatenated tableaux of the rows in $B^{(j)}$. The expected size of $B^{(j)}$ is at most $\eta \cdot |U^{(j)}|$, where $\eta = \widetilde{O}(1/2^{3\sqrt{\log T}})$. By Markov's inequality, $|B^{(j)}| \leq |U^{(j)}|/2$ with probability at least $1 - 2\eta = 1 - o(1)$ in each iteration.

Assume that $|B^{(j)}| \leq |U^{(j)}|/2$ (the non-degenerate case). For each surviving row, which induces a tableau of length $\ell^{(j)}$, the prover also sends the mid-point of that computation. This yields a new table $U^{(j+1)}$ of size $2|B^{(j)}| \leq |U^{(j)}|$. The new computation length is $\ell^{(j+1)} = \ell^{(j)}/2$. The new predicate $\phi^{(j+1)}$ checks that:

1. the new configurations are the correct mid-points of their respective computations, and
2. the output predicate $\psi^{(j)}$ from the previous iteration accepts the concatenated tableaux of all the rows, and
3. the initial state of each tableau given to $\psi^{(j)}$ equals the corresponding row of $U^{(j+1)}$.

Since $\psi^{(j)}$ is $2^{O(\sqrt{\log T^{(j)}})}$-succinct and has depth $2^{O(\sqrt{\log T^{(j)}})}$, the predicate $\phi^{(j+1)}$ also satisfies the depth and succinctness requirements of Theorem 5.4.

If instead $|B^{(j)}| > |U^{(j)}|/2$ (the degenerate case), the prover and verifier ignore the result of the honing protocol and carry over the same table $U^{(j+1)} = U^{(j)}$, the same predicate $\phi^{(j+1)} = \phi^{(j)}$, and halve the computation length to $\ell^{(j+1)} = \ell^{(j)}/2$. We call such an iteration *ignored*. As argued above, each iteration is ignored with probability at most $2\eta = o(1)$. If more than half of the $r = O(\log T)$ iterations are ignored, the verifier aborts and accepts.

Final iteration. The probability that more than half of the $r = O(\log T)$ iterations are ignored is at most $1/\mathsf{poly}(T)$ by a Chernoff bound (since each is ignored independently with probability $2\eta = o(1)$ This probability depends only on the verifier's coins). Conditioned on this not happening, after $r = O(\log T)$ iterations the computation length has been halved at least $r/2$ times, giving $\ell^{(r)} \leq T/(k \cdot 2^{r/2}) = \mathsf{polylog}(T)$. The final table $U^{(r)}$ has $\mathsf{polylog}(T)$ size. The verifier can therefore evaluate the final predicate $\phi^{(r)}$ on its own in time $\mathsf{polylog}(T)$.

Completeness and soundness. Completeness follows by design: an honest prover sends the correct configurations and tableaux, and every predicate is satisfied.

For soundness, first observe that the probability of too many ignored iterations is at most $1/\mathsf{poly}(T)$ as argued above; conditioned on this not happening, the final predicate is reached and the verifier evaluates it directly.

If the prover's initial claim is false, then the correct tableau leads to a rejecting state, so $\phi^{(1)}$ is not satisfiable, and $U^{(1)}$ is at infinite distance from any table whose induced tableaux make $\phi^{(1)}$ accept. In each non-ignored iteration j, if $\phi^{(j)}$ is not satisfiable then the input table is at infinite distance from a table making $\phi^{(j)}$ accept, and by the soundness of the honing protocol the output predicate $\psi^{(j)}$ rejects the tableaux of $B^{(j)}$ with probability at least $1 - \varepsilon_h$. In that case, since $\phi^{(j+1)}$ checks that the initial states equal $U^{(j+1)}$ and that the new configurations are correct midpoints, $\phi^{(j+1)}$ is also not satisfiable. For an ignored iteration, $\phi^{(j+1)} = \phi^{(j)}$ is also not satisfiable by assumption.

By a union bound over the $r = O(\log T)$ non-ignored iterations, with probability at least $1 - r \cdot \varepsilon_h = 1/2$ the final predicate is not satisfiable and the verifier rejects. The soundness error is reduced to ε by $O(\log(1/\varepsilon))$ parallel repetitions.

Complexity. The predicates in all iterations satisfy the conditions of Theorem 5.4: they are $\mathsf{poly}(T)$-size, $2^{O(\sqrt{\log T})}$-depth, and $\rho^{(j)}$-succinct with $\rho^{(j)} \leq 2^{O(\sqrt{\log T})}$. In each iteration, the honing protocol uses communication $S \cdot 2^{O(\sqrt{\log T})} \cdot \log(1/\varepsilon_h)$, rounds $2^{O(\sqrt{\log T})}$, and verifier time $(|x| + S) \cdot 2^{O(\sqrt{\log T})} \cdot \log(1/\varepsilon_h)$. The prover runs in $\mathsf{poly}(T)$ time per iteration. There are $O(\log T)$ iterations, and claimed complexity bounds follow. □

Acknowledgments

We thank Noga Amit, Oded Goldreich and Ron Rothblum for helpful and illuminating discussions on these topics.

A The AR Argument System

We sketch the construction and proof behind the HIA of Theorem 3.24, we refer to this protocol as the AR argument system [AR23] throughout this appendix.

Layer-by-layer commitment. We view the circuit as a layered circuit with fan-in 2. The AR argument begins with the prover committing to the LDE of each layer in the circuit: we number the layers from 1 (the top gate) to D (the input). Fixing the explicit and holographic inputs, let $V_i \in \{0,1\}^S$ be the vector of values of the gates at layer i on that input. Let $\widehat{V}_i \in \mathbb{F}^{\mathsf{poly}(S)}$ be the LDE of the i-th layer (which we view as a string or as a function in the usual way). For now we take $|\mathbb{H}| = \mathsf{polylog}(S)$, $|\mathbb{F}| = \mathsf{poly}(|\mathbb{H}|)$, see Remark A.1 for working with general field sizes.

Remark A.1 (General field sizes)**.** *The construction is described w.r.t fields $\mathbb{H}$ and $\mathbb{F}$ of sizes that are polylogarithmic in S. We want the protocol to support general field sizes, e.g. because this is needed for compatibility with the* PVAL IPP *of [RR20]. We can move from* LDE *claims over fields $\mathbb{H}, \mathbb{F}$ to new fields $\mathbb{H}', \mathbb{F}'$ by running the GKR protocol as a holographic proof w.r.t. $\mathbb{H}', \mathbb{F}'$ on the circuit that computes $\mathsf{LDE}_{\mathbb{H},\mathbb{F}}$. This is a log-space uniform* NC *circuit, so the communication, round complexity and (holographic) verifier runtime are poly-logarithmic in the length $n \leq S$ of the input being encoded. For $|\mathbb{H}|' \leq \mathsf{polylog}(S)$, the soundness error of the GKR protocol is $(\mathsf{polylog}(S)/|\mathbb{F}'|)$. The soundness error of the original* HIA *is $1/\mathsf{polylog}(S)$: if needed, this can be reduced by repetition (where we then use the GKR protocol to verify all the claims about $\mathsf{LDE}_{\mathbb{H},\mathbb{F}}$ in these repetitions), giving the soundness bound claimed in Theorem 3.24.*

The verifier sends to the prover a key (hash functions) for a the succinct locally openable commitment scheme, and the prover responds with a commitment y_i for each layer. The commitment to the LDE of the i-th layer is computed using a UOWHF tree, as described in the overview and in Section 3.1. The original AR protocol aimed for constant round complexity, and used a constant-depth hash tree with polynomial arity. Since we only aim for $\mathsf{polylog}(S)$ rounds, we can use a binary hash tree of logarithmic depth. The commitment y_i is the root of this hash tree.

To begin, the verifier asks the prover to open the succinct commitment to the output layer and checks that the circuit's output gate outputs 1. The main technical tool used in the protocol is a *holographic hash root* (HHR) protocol: the verifier in this protocol takes the alleged hash root y_i as an explicit input, and outputs a claim about a single coordinate in the LDE of layer $(i+1)$. If y_i was not the correct hash root for $\widehat{V}_i$, then w.h.p. the claim about $\widehat{V}_{i+1}$ will also be false (this soundness guarantee is unconditional).

The prover and the verifier run the HHR protocol in parallel for each layer $i \in [D-1]$, resulting in holographic claims $((z_i, v_i))_{i\in[2,D]}$. For each $i \in [2, D-1]$, the verifier also asks the prover to open the commitment y_i at location z_i and verifies that the opening is to the value v_i. If all these tests pass, then the verifier outputs the claim (z_D, v_D) about the LDE of the input.[12]

For soundness: if the hash root for the next-to-last layer was corrupted (i.e. not the correct hash root for $\widehat{V}_{D-1}$), then the claim (z_D, v_D) about the input layer will be false and the verifier indeed outputs a false claim about the LDE of the input. Otherwise, since we know that the root

[12] If there is an explicit input, then we decompose the claim about the input layer into claims about the holographic and the explicit parts, see Proposition 3.20. The prover sends alleged values for both these parts, the verifier checks that the decomposition is valid, explicitly checks that the explicit part is true, and if so then it outputs the prover's claim about the holographic part.

y_1 was corrupted (it opens to an accepting output wire, whereas in the soundness proof we assume that the circuit rejects the input), for some $i \in [D-2]$, we have that the commitment to layer i is a corrupted hash root, but the commitment to layer $(i+1)$ was correct. Crucially, the committed values $(\widehat{V}_i)_{i\in[D]}$ were fixed *before* the verifier sent its hash keys, so the correct hash roots are binding. Now in the i-th HHR execution, since the root y_i was corrupted, the resulting holographic claim (z_{i+1}, v_{i+1}) about the LDE of the $(i+1)$-th layer will be false. However, the prover's commitment on the $(i+1)$-th layer is binding: it cannot open that commitment to the alleged value, and the verifier will reject (unless the prover breaks the targeted collision-resistance of the UOWHF).

Before proceeding to describe the HHR protocol, we decompose it into two parts: reducing from the claim about the hash root of layer i to a claim about the LDE of layer i, and then reducing from a claim about the LDE of layer i, to a claim about the LDE of layer $(i+1)$. In our setting, the latter part can be performed in $\mathsf{polylog}(S)$ rounds, communication and verifier runtime using the GKR protocol (whereas the parameter setting of [AR23] needed a constant-round protocol for this part). We thus focus on the first part: an HHR protcol that verifies the hash root of a layer and outputs a claim about the LDE of that same layer.

Holographic hash root (HHR) protocol. So far, we followed the template of the AR protocol, except for our use of a binary hash tree (rather than a tree with arbitrarily small polynomial arity). The HHR protocol of [AR23] was designed to work with hash trees of constant depth, and in constant round complexity. We use the same construction, but for trees of logarithmic depth and with polylogarithmic round complexity. The advantage is that the communication complexity and verifier runtime are reduced from $n^{\sigma} \cdot \mathsf{poly}(\kappa)$ to $\mathsf{polylog}(S) \cdot \mathsf{poly}(\kappa)$. The details follow.

We present an unconditionally sound holographic IP that verifies the hash root y of an LDE $\widehat{V}$, and outputs a single claim about that LDE. The AR protocol follows the computation of the hash tree. Taking M to be the length of the LDE (viewed as a binary string), we assume w.l.o.g. that M is a power of 2 and take $m = \log_2 M$ to be the depth of the hash tree. We assume that the UOWHFs map string of length 2κ to strings of length κ. In the i-th layer of the hash tree there are 2^i nodes, each of size κ. Let $W_i \in \{0,1\}^{2^i \cdot \kappa}$ be the concatenation of the values of the nodes at layer i of the tree. Thus, treating each bit of the input layer as a node, we have the "tree input" layer $W_m = \widehat{V}$ and the "tree output" layer $W_0 = y$.

The HHR protocol proceeds by sequential applications of a *tree-layer subprotocol* to reduce from verifying a claim about $\mathsf{LDE}(W_i)$ to a claim about $\mathsf{LDE}(W_{i+1})$. We detail this protocol below: in our setting of parameters, there are $m = \log(|\widehat{V}|) = O(\log(S))$ sequential applications of this protocol within each of the $(D-1)$ HHR protocols that are performed in parallel.

There are some subtleties in dealing with the bottom layer of the hash tree: we don't want to end up with a claim about the LDE of (the binary representation of) an LDE. These are handled in the AR protocol by viewing the input to the hash tree as the unencoded string V, and including the encoding of that string in the computation, see [AR23, Remark A.2]. This is easy to extend to our parameter setting and we ignore this subtlety here.

Tree-layer sub-protocol. In the i-th tree layer subprotocol call we consider the i-th layer of the hash tree, which is computed using a hash function h (a different function is used for each layer, see Construction 3.5). Fixing the input $\widehat{V}$, in the i-th layer we have an output $W_i = (y_1, \ldots y_{2^i})$ where each $y_j \in \{0,1\}^{\kappa}$, and an input $W_{i+1} = (z_1, \ldots z_{2^i})$, where each $z_j \in \{0,1\}^{2\kappa}$. We want a holographic proof system that verifies the input claim about $\mathsf{LDE}(W_i)$ and also that $\forall j \in [2^i],\ y_j = h(z_j)$. The

key insight in this part of the AR protocol is that this is a batch-verification problem, and it can be verified using a protocol similar to the UP verification protocols of [RRR18, RR20]. The idea is to run in iterations, where in each iteration $\ell \leftarrow k \ldots i' \leq i$:

- At the beginning of the k-th iteration, we have a claim about a subset $S^k \subseteq [2^i]$ of the y_j values. The claim is an easy-to-compute predicate ϕ^k that is claimed to hold about the values $(y_j)_{j \in S^k}$. Initially, the set $S^1 = [2^i]$ includes all the rows, and the predicate ϕ^1 checks that the claim about the LDE of W_i is true.

- The prover and verifier run an output-predicate IPP (Definition 3.26): the implicit input is $(y_j)_{j \in S^k}$ and the IPP also takes the entire $W_{i+1} = (z_1, \ldots, z_{2^i})$ as a holographic input. It checks that $\phi^k((y_j)_{j \in S^k}) = 1$ and that for every $j \in S^k$, $y_j = h(z_j)$. The IPP outputs a holographic claim about the LDE of W_{i+1} and an output predicate ϕ^{k+1} about a subset of the locations in the implicit input $(y_j)_{j \in S^k}$. We take S^{k+1} to be the set of y_j's that are read by the IPP's output predicate, and the parameters of the IPP are set so that $|S^{k+1}| \leq |S^k|/2$.

  Fixing the holographic input W_{i+1}, if the prover's claim was false, then there is no implicit input satisfying the predicate (if the implicit input is the one induced by the holographic input, then it doesn't satisfy ϕ^k). As in the UP batching protocol of [RR20], this means that w.h.p. either the holographic output claim is false or the output predicate ϕ^{k+1} is false on the subset S^{k+1} of inputs.

  Using the IPP of [RR20], for each iteration the communication complexity and verifier runtime are $\mathsf{poly}(\kappa, \log(S))$. The main distinction from the parameter setting of [AR23], is that they could only afford a constant number of iterations, so the size of S^k needed to be reduced by a polynomial factor in each iteration, which required polynomial communication complexity and verifier runtime. We remark that in the full-blown AR construction (and here too), the implicit input to the IPP also needs to include the tableau of the computation of h on each $(z_j)_{j \in S^k}$. This is done so the depth of the predicate ϕ^k does not depend on the depth of computing h and does not meaningfully change the parameters.

The iterations end at iteration $i' = O(\log(S))$, when $|S^{i'}| = \mathsf{polylog}(S)$, at which point the prover can sent the explicit values $(y_j)_{j \in S^{i'}}$ to the verifier. The verifier checks that $\phi^{i'}$ accepts these values, and runs a final holographic proof to verify that they are consistent with W_{i+1}. Finally, the verifier "combines" the $(i'+1)$ claims it accrued about the LDE of W_{i+1} into a single holographic claim that it outputs (the combination is done, as usual, by passing a curve through all the points, asking the prover to interpolate the composition of the LDE with the curve, and then having the verifier choose a random point on the interpolated curve).

References

[AR23] Noga Amit and Guy N. Rothblum. Constant-round arguments from one-way functions. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1537–1544, New York, NY, USA, 2023. Association for Computing Machinery.

[AR24] Noga Amit and Guy N. Rothblum. Constant-round arguments for batch-verification and bounded-space computations from one-way functions. In Leonid Reyzin and Douglas

Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 3–37. Springer, 2024.

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

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

[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.

[BGHK25] Bonnie Berger, Rohan Goyal, Matthew M. Hong, and Yael Tauman Kalai. Efficiently batching unambiguous interactive proofs. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 818–863. IEEE, 2025.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, October 19-22, 1997*, pages 374–383. IEEE Computer Society, 1997.

[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.

[CHKX26] Liyan Chen, Matthew M. Hong, Yael Tauman Kalai, and Zoe Xi. Towards a doubly efficient ip=pspace. *ECCC*, TR26-102, 2026.

[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.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004.

[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.

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

[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.

[GRS23] Oded Goldreich, Guy N. Rothblum, and Tal Skverer. On interactive proofs of proximity with proof-oblivious queries. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, MIT, Cambridge, Massachusetts, USA, January 10-13, 2023*, LIPIcs, pages 59:1–59:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Hai13] Iftach Haitner. A parallel repetition theorem for any interactive argument. *SIAM J. Comput.*, 42(6):2487–2501, 2013.

[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.

[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.

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

[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.

[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.

[RR20] Guy N. Rothblum and Ron D. Rothblum. Batch verification and proofs of proximity with polylog overhead. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 108–138. Springer, 2020.

[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.

[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.

[Wol65] Jack K. Wolf. On codes derivable from the tensor product of check matrices. *IEEE Trans. Information Theory*, 11(2):281–284, 1965.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il