Electronic Colloquium on Computational Complexity, Report No. 102 (2026)

Towards a Doubly Efficient IP = PSPACE

Liyan Chen[1], Matthew M. Hong[1], Yael Tauman Kalai[1], and Zoe Xi[1]

[1]Massachusetts Institute of Technology

June 17, 2026

Abstract

We show that every language in PSPACE decidable by a Turing machine in time $T(n) = n^{O(\log n)}$ admits a doubly efficient interactive proof system: the prover runs in time $\mathsf{poly}(T(n))$, and the verifier runs in time $\mathsf{poly}(n)$. This extends the best previously known regime for such proof systems from $T(n) = n^{O(\sqrt{\log n / \log\log n})}$, established by Berger, Goyal, Hong, and Kalai (FOCS 2025), to $T(n) = n^{O(\log n)}$.

Beyond improving the range of T, our protocol is substantially simpler than previous doubly efficient proofs for time-bounded PSPACE. Earlier constructions proceed indirectly: they first build batch interactive proofs and then invoke them as a black box to obtain doubly efficient protocols. In contrast, we give a direct construction. This not only simplifies the proof but also points to a more promising route for future improvements.

ISSN 1433-8092

Contents

1 Introduction

Verification is one of the most fundamental concepts in computer science, and motivated the study of the complexity classes NP [Coo71], IP [GMR89, BM88] and MIP [BGKW88, BFL90], among others. This study has led to foundational concepts in cryptography such as zero-knowledge proofs [GMR89], which are used as building blocks in many cryptographic protocols used today. It has also inspired fundamental notions such as probabilistically checkable proofs (PCPs) [BFLS91, FGL+91, AS92, ALM+92], interactive PCPs [KR08], and interactive oracle proofs [BCS16, RRR16], which in turn have led to breakthrough results in hardness of approximation and to the construction of SNARGs (succinct non-interactive arguments), which are used in many blockchain applications.

Interactive proofs (IPs) The power of interactive proofs was demonstrated by the celebrated IP = PSPACE theorem [LFKN92, Sha92]. Specifically, it was proven that the correctness of any time-T, space-S computation can be verified by a $\mathrm{poly}(S, n)$-time verifier, via an interactive proof. However, this comes at a price: The time required by the prover to convince the verifier of the correctness is $2^{\Omega(S \cdot \log S)}$. While in those works the runtime of the prover was not a parameter of interest, and the prover was thought of as being all-powerful (and was even named after the famous wizard Merlin [BM88]), this blowup in the prover's runtime makes these proof systems completely impractical.

Doubly efficient IPs The work of [GKR15] initiated the study of *doubly efficient* IPs, in which the prover's runtime is required to be at most polynomial in T (the time it takes to run the underlying computation), and the verifier is required to run in time significantly less than T (otherwise, such interactive proofs are trivial). They constructed a doubly efficient IP for any computation that can be performed by a (log-space uniform) circuit of depth D and size T, where the verifier's runtime is $D \cdot \mathrm{polylog}(T) + \tilde{O}(n)$, and the communication complexity is $D \cdot \mathrm{polylog}(T)$, where n is the input length. In particular, this implies an improved IP = PSPACE theorem where the verifier runs in time $\mathrm{poly}(S) + \tilde{O}(n)$, the communication complexity is $\mathrm{poly}(S)$, and the prover runs in time $2^{O(S)}$. This result yields a doubly efficient IP for log-space computations.

The next significant advancement was due to Reingold, Rothblum, and Rothblum [RRR16], who constructed a doubly efficient IP, where the prover's runtime is $\mathrm{poly}(T)$ and the verifier's runtime is $\mathrm{poly}(n)$, for every language computable in polynomial space and time $T = n^{O((\log n)^{\delta})}$ for a sufficiently small constant $\delta > 0$. Very recently, [BGHK25] improved this result by constructing a doubly efficient IP for every language in polynomial space and time $T = n^{O\left(\sqrt{\log n / \log \log n}\right)}$.

The main technical contribution in both of these works is an efficient way to batch unambiguous interactive proofs. An unambiguous interactive proof is an interactive proof with the guarantee that if the prover ever deviates from the unique prescribed strategy, then it will be rejected with high probability. Both results mentioned above [RRR16, BGHK25] show that if membership in a language $\mathcal{L}$ can be proven via a doubly efficient unambiguous IP, then one can prove that $x_1, \ldots, x_k \in \mathcal{L}$ via a doubly efficient unambiguous IP where the verifier runs in time significantly less than running these k proofs. These works then use this batch unambiguous IP in a black-box way to construct a doubly efficient IP. This latter part is quite straightforward. Indeed, the improvement in [BGHK25] is in constructing a more efficient batch unambiguous IP, which immediately yields a more efficient doubly efficient IP.

1.1 Our Result

We construct a doubly efficient IP for every language that is computable in polynomial space and time $T = n^{O(\log n)}$, thus improving both previous results.

Theorem 1 (Our Doubly Efficient IP, Informal)**.** *There exists a doubly efficient IP (i.e. an interactive proof with a* $\mathrm{poly}(T)$ *prover and a* $\mathrm{poly}(n)$ *verifier) for every language that is computable in polynomial space and time* $T = n^{O(\log n)}$*.*

More generally, we prove the following theorem.

Theorem 2 (Our General Doubly-Efficient IP, Informal)**.** *There exists an IP for every language that is computable in time* T *and space* S*, where the prover runs in time* $\mathrm{poly}(T)$ *and the verifier runs in time* $\mathrm{poly}(S, 2^{\sqrt{\log T}})$*.*

Our construction is significantly simpler than the constructions in both of these prior works. In particular, the construction is direct. As opposed to previous works, which use a batch unambiguous IP as a black box, we directly batch DTISP computations. We believe that this direct construction has the potential to lead to further improvements, while constructions that use batch (unambiguous) IP as a black box are subject to barriers. Specifically, to get a doubly efficient IP for languages computable in time $T = n^{\omega(\log n)}$ and polynomial space using a batch (unambiguous) IP as a black box, one would need a "rate-1" batch (unambiguous) IP, i.e., one where the communication complexity for proving k statements is $(1 + o(1)) \cdot \mathsf{cc}$, where cc is the communication complexity of proving a single statement. Constructing such a rate-1 batch (unambiguous) IP seems challenging. On the other hand, we believe that our protocol points to a more promising route for future improvements.

1.2 Additional Related Work

Our protocol builds on two sub-protocols from prior works. The first is the doubly efficient IP for bounded depth circuits from [GKR15] which we mentioned above. The second is an "instance reduction" protocol from [RRR16, RRR18, RR20], which roughly says that if $x_1, \ldots, x_k \in \mathcal{L}$ is far from being true (i.e., one needs to change, say d, of the statements x_i to make it true), then there is a so-called "instance reduction" protocol that reduces checking that $x_1, \ldots, x_k \in \mathcal{L}$ to checking that $\tilde{O}(k/d)$ of these instances are in $\mathcal{L}$ (with an additional linear test on these $\tilde{O}(k/d)$ instances). Such an instance-reduction protocol is special case of an interactive proof of proximity (IPP) [RVW13], which is an efficient protocol that, given an input x far from satisfying some predicate P, i.e., one needs to change at least d coordinates in x to satisfy P, reduces verifying that $P(x) = 1$ to verifying that $P'(x_S) = 1$, where P' is a related predicate and S is a subset of indices of size $\tilde{O}(|x|/d)$.

We also mention that there is a large body of work, starting with [BCC88, Mic94], on constructing computationally sound proofs, where soundness is guaranteed to hold only against computationally bounded cheating provers. We do not elaborate on these works here, since they are less relevant for our work.

2 Technical Overview

In this section, we give an overview of our approach for proving membership in a $\mathsf{TISP}(T, S)$ language, where the verifier is given a Turing machine M and two configurations cf_0 and cf_T of size bounded by S, and it wishes to verify whether running M for T steps from cf_0 reaches cf_T.

One natural idea is for the prover to send intermediate configurations: let λ be a fixed parameter,[1] and the prover sends

$$\mathsf{cf}_{T/\lambda}, \mathsf{cf}_{2T/\lambda}, \ldots, \mathsf{cf}_{(\lambda-1)T/\lambda}$$

as the λ intermediate configurations after $\frac{T}{\lambda}, \frac{2T}{\lambda}, \ldots, \frac{(\lambda-1)T}{\lambda}$ steps, respectively. This reduces the task of verifying a T-time statement $(\mathsf{cf}_0 \xrightarrow{T} \mathsf{cf}_T)$ to the task of verifying λ many T/λ-time statements

$$\mathsf{cf}_{i\cdot T/\lambda} \xrightarrow{T} \mathsf{cf}_{(i+1)\cdot T/\lambda} \quad \text{for all} \quad 0 \leq i < \lambda.$$

Prior works, starting with [RRR16], and continuing with [RRR18, RR20, BGHK25], showed that verifying k statements $x_1, \ldots, x_k$ is easier than verifying each of them separately. Roughly speaking, this is based on the following initial observation: if we are guaranteed that a constant fraction of $x_1, \ldots, x_k$ are false, then it suffices to randomly sample a small subset of $x_1, \ldots, x_k$ and perform verification on that subset. Following this observation, previous works on batching (unambiguous) NP statements [RRR18, RR20] and batching (unambiguous) interactive proofs [RRR16, BGHK25] manage to cleverly reduce the number of instances by a constant factor, even when initially only a single claim is false, by running sub-protocols that add sufficiently many constraints on the corresponding witnesses $(w_1, \ldots, w_k)$, or on the transcripts in the case of (unambiguous) IP, such that with these additional constraints many of the statements become false. They then repeat this recipe until they are left with a constant number of instances, which the verifier can verify on its own or via an interactive proof.

In order to obtain a doubly efficient interactive proof for $\mathsf{TISP}(T, S)$, previous works, as well as this work, first break down the statement into λ statements, each in $\mathsf{TISP}(T/\lambda, S)$. They then apply a batching protocol for these λ statements, reducing the task to verifying a constant number of statements in $\mathsf{TISP}(T/\lambda, S)$. These works then break down each of the remaining statements in $\mathsf{TISP}(T/\lambda, S)$ into λ statements, each in $\mathsf{TISP}(T/\lambda^2, S)$, and apply the batching protocol again. This recipe is repeated until $T = O(1)$ (or until it is polynomial), in which case the verifier can verify the statements on its own.

What differentiates this work from previous works is the batching protocol used. Previous works used a batch protocol for unambiguous *interactive proofs*, which is conceptually complex and brings a lot of technical difficulties. This work, on the other hand, directly batches $\mathsf{TISP}(T, S)$ statements. Our batching protocol recursively reduces both k and T, and is surprisingly simple.

Core Ingredient: Interactive Proof of Proximity Before describing our protocol, we first introduce an important ingredient: an Interactive Proof of Proximity (IPP).

An IPP is a proof system that combines interactive proofs with property testing: a sublinear-time verifier, who can only query a few bits of a huge input x, interacts with a prover to decide whether x satisfies some property Π or is far (in Hamming distance) from every string that satisfies property Π. Completeness requires that if x satisfies property Π, then the honest prover convinces the verifier to accept with probability 1. Soundness requires that if x is far from satisfying Π in Hamming distance, no prover strategy will make the verifier accept with more than a small probability. Previous work [RR20] constructs an IPP for any low-depth property with communication complexity $\tilde{O}(d)$ and query complexity $\tilde{O}(n/d)$, where d is the distance threshold corresponding to the soundness guarantee. Furthermore, the IPP verifier makes *non-adaptive* queries: it never queries the instance

[1] We will later discuss how to set this parameter.

x until the end of the interaction, at which point it samples a random set of query indices Q (depending on its randomness in previous rounds and this end phase), and the final decision is a low-depth predicate that depends only on x_Q.

This work uses a generalization of IPP from [BGHK25], called row-IPP, which treats the huge input x as a large matrix M with many rows. The verifier can only query a few rows in M, and soundness holds for any M with a large row-distance from the property Π, i.e., for any M for which every M' that satisfies Π differs from M in many rows. Row-IPP is known for parameters similar to those of standard IPP; specifically, for $n \times m$ matrices and distance parameter d, [BGHK25] adapts the IPP from [RR20] into a row-IPP for any low-depth property with communication complexity $\tilde{O}(dm)$ and query complexity $\tilde{O}(n/d)$. The row-IPP verifier also makes *non-adaptive* queries: the final decision is a low-depth predicate that depends on a few rows in M.

Batching TISP: A 4-Step Construction. Our protocol $\mathsf{Batch}(T, k)$ for proving k $\mathsf{TISP}(T, S)$ statements is as follows. Assume that we are proving k statements $(x_1, \ldots, x_k)$, where

$$x_i : \mathsf{cf}_0^i \xrightarrow{T} \mathsf{cf}_T^i$$

is a time-T computation. An immediate idea is to apply the famous GKR protocol [GKR15] to verify the correctness of all k $\mathsf{TISP}(T, S)$ statements. However, the verification time of the GKR protocol depends on the *depth* of the computation, which is $O(T\mathrm{poly}(S)\log k)$ for the union of k $\mathsf{TISP}(T, S)$ statements: The main overhead for applying GKR is T. Our main motivation for the following steps is to reduce T of the TISP statements we need to verify.

We let w_i be the λ intermediate configurations, i.e.,

$$w_i = (\mathsf{cf}_{T/\lambda}^i, \mathsf{cf}_{2T/\lambda}^i, \ldots, \mathsf{cf}_{(\lambda-1)T/\lambda}^i).$$

Consider the following k-by-λ matrix, denoted by M, where the i-th row consists of w_i, and to simplify exposition, we also include in the i-th row the statement $x_i = (\mathsf{cf}_0^i, \mathsf{cf}_T^i)$ in the first and last columns, respectively:

$$M = \begin{pmatrix} \mathsf{cf}_0^1 & \mathsf{cf}_{T/\lambda}^1 & \mathsf{cf}_{2T/\lambda}^1 & \cdots & \mathsf{cf}_{(\lambda-1)T/\lambda}^1 & \mathsf{cf}_T^1 \\ \mathsf{cf}_0^2 & \mathsf{cf}_{T/\lambda}^2 & \mathsf{cf}_{2T/\lambda}^2 & \cdots & \mathsf{cf}_{(\lambda-1)T/\lambda}^2 & \mathsf{cf}_T^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathsf{cf}_0^k & \mathsf{cf}_{T/\lambda}^k & \mathsf{cf}_{2T/\lambda}^k & \cdots & \mathsf{cf}_{(\lambda-1)T/\lambda}^k & \mathsf{cf}_T^k \end{pmatrix}.$$

The matrix M defines $k\lambda$ $\mathsf{TISP}(T/\lambda, S)$ statements, by taking each pair of consecutive elements in a row as the start and end configurations. We denote the $k\lambda$ statements defined by M as $\boldsymbol{x}_M$.

Let λ, d be two global parameters. The protocol $\mathsf{Batch}(T, k)$ is as follows:

1. **Sending Checksums.** The prover computes checksums of M, denoted by cksum, such that any two matrices that are consistent with cksum have row-distance at least $(2d+1)$ (i.e., they differ in at least $(2d+1)$ rows). It sends cksum to the verifier.

2. **Reducing T.** The prover and verifier engage in $\mathsf{Batch}(T/\lambda, k\lambda)$ over $\boldsymbol{x}_M$. However, we delay the verification of $\mathsf{Batch}(T/\lambda, k\lambda)$: the verifier does not access $\boldsymbol{x}_M$ directly; instead, it outputs a low-depth predicate ψ about $\boldsymbol{x}_M$ such that $\psi(\boldsymbol{x}_M) = 0$ if $\boldsymbol{x}_M$ is not correct. Define Ψ as $\Psi(M) = \psi(\boldsymbol{x}_M)$.

Remark 1. The purpose of these two steps is to create distance: Either every M' that is consistent with cksum is d-far from M, or there exists an M' consistent with cksum that is d-close to M for which we will show $\Psi(M') = 0$.

3. **IPP.** The prover and verifier engage in a row-IPP proving that M is close in row-distance to satisfying the following property (checkable by a low-depth circuit):

   - M is consistent with cksum;
   - $\Psi(M) = 1$.

   The distance parameter of this row-IPP is d, and the verifier at the end needs to query k/d rows $M[\mathcal{S}, :]$ in M. Note that $M[\mathcal{S}, :]$ are the λ intermediate configurations of $\{x_i : i \in \mathcal{S}\}$. The output of the IPP verifier is a low-depth predicate Φ about $M[\mathcal{S}, :]$.

4. **Reducing k.** The prover and verifier engage in $\mathsf{Batch}(T, k/d)$ over k/d $\mathsf{TISP}(T, S)$ statements $\{x_i : i \in \mathcal{S}\}$, with the goal of checking that $M[\mathcal{S}, :]$, defined by the k/d statements $\{x_i : i \in \mathcal{S}\}$, satisfies Φ, and that the k/d statements $\{x_i : i \in \mathcal{S}\}$ are valid. We recurse over a batched protocol $\mathsf{Batch}(T, k/d, \Phi)$ that checks a low-depth predicate Φ on the *middle configurations* and follows exactly the same recursive construction that we are describing. In this technical overview, we forget Φ for simplicity, and only say we want to check if all of $\{x_i : i \in \mathcal{S}\}$ are valid.

The base case is when $T = O(1)$ or $k = O(1)$.

- T **is small**. In this case, all k $\mathsf{TISP}(T, S)$ statements can be verified by a low-depth circuit. Thus, the prover and verifier can engage in the GKR protocol, which returns a low-degree extension check on the statements.
- k **is small**. In this case, the prover sends the entire M in the clear, and the prover and verifier engage in $\mathsf{Batch}(T/\lambda, k\lambda)$.

The above is the full description of our construction. We analyze it and provide concrete parameters below.

Protocol Analysis: Win-Win Argument In what follows, we say that the "unique" witness (or true configurations) corresponding to the i-th statement $x_i = (\mathsf{cf}_0^i, \mathsf{cf}_T^i)$ consists of the middle configurations obtained by computing honestly from cf_0^i (even if the statement x_i is false). Thus, M is well-defined even if some statements are false. Our analysis goes through the four steps in the protocol.

Step 1: Sending Checksums. In the first step of the protocol, the prover sends checksums, denoted by cksum, for all the columns of M. The size of each column of cksum is set to be $\tilde{O}(d)$ so that one can (uniquely) decode any deviation in at most d rows. It is convenient to think of $d = \lambda$ since we will indeed set these two parameters to be equal. The total size of cksum is $\tilde{O}(d) \cdot |w_i| = \tilde{O}(d \cdot \lambda \cdot S)$, where S is the size of each configuration.

We distinguish between two cases, both of which take us a step closer to catching the cheating prover:

- **Case 1:** One needs to change more than d rows of the unique witnesses M to be consistent with cksum. In this case, M already has a large row-distance to the property proven by row-IPP in step 3. This is an easy case as we can use the soundness guarantee of the row-IPP. In this case step 2 of the protocol is not necessary since we already have the distance needed for the IPP soundness.
- **Case 2:** One needs to change fewer than d rows of the unique witnesses M to be consistent with cksum. In this case, we cannot directly say that the distance condition for row-IPP soundness is satisfied. However, this case still offers us a win: cksum uniquely determines the witnesses sent by the prover, since for each column, cksum uniquely decodes up to d deviations. In other words, if we haven't created distance then the prover has committed to a matrix M' that is d-close to M through cksum.

Note that the committed matrix M' is a fixed k-by-λ matrix that contains $k\lambda$ configurations, corresponding to $k\lambda$ statements $\boldsymbol{x}_{M'}$, where each statement corresponds to a T/λ-time computation. We can assume that M' and M have the same starting and ending configurations in each row, since the verifier can efficiently check this. This implies that one of these $k\lambda$ statements in $\boldsymbol{x}_{M'}$ must be false.

Step 2: Reducing T. The purpose of this step is to create the distance needed for the IPP if we are in Case 2 in the last step. At step 2, the prover and verifier run the protocol $\mathsf{Batch}(T/\lambda, k\lambda)$. Recall that the verifier does not access the instance, but only outputs a predicate ψ about this instance. In our soundness analysis, we regard this $\mathsf{Batch}(T/\lambda, k\lambda)$ as executed w.r.t. $\boldsymbol{x}_{M'}$, the $k\lambda$ statements defined by M'. Therefore, by soundness of $\mathsf{Batch}(T/\lambda, k\lambda)$, with high probability $\psi(\boldsymbol{x}_{M'}) = 0$, i.e. $\Psi(M') = 0$.

Step 3: IPP. At step 3, the prover and verifier engage in a row-IPP proving that M is close in row-distance to the property defined by the cksum from step 1 and Ψ from step 2. Recall that in step 1 we distinguished between two cases: M is close (in row distance) to cksum or not. We next claim that in both cases M is far (in row distance) from the property that row-IPP is proving:

- **Case 1:** M is already far from the satisfying checksums.
- **Case 2:** The only possible matrix close to M and satisfying the checksums is M'. However, $\Psi(M') = 0$.

Therefore, by soundness of row-IPP, with high probability, the verifier obtains a set $\mathcal{S}$ and predicate Φ such that $\Phi(M[\mathcal{S}, :]) = 0$.

Step 4: Reducing k. We are left with checking $\Phi(M[\mathcal{S}, :]) = 1$. If we don't need to check Φ but need only check whether $\{x_i : i \in \mathcal{S}\}$ are correct, then simply running $\mathsf{Batch}(T, k/d)$ is sufficient. In our actual construction, we instead check if Φ is satisfied by the intermediate configurations $M[\mathcal{S}, :]$ uniquely determined by $\{x_i : i \in \mathcal{S}\}$. This turns out to be extremely close to checking correctness of $\{x_i : i \in \mathcal{S}\}$, and we recursively build $\mathsf{Batch}(T, k, \Phi)$ for the above condition with the same 4-step construction.

Efficiency analysis In the construction above, the $\mathsf{Batch}(T, k)$ protocol runs a $\mathsf{Batch}(T/\lambda, k\lambda)$ protocol and a $(T, k/d)$ protocol with additional $\tilde{O}(\lambda \cdot d \cdot S)$ communication and $\mathrm{poly}(\lambda, d, S)$ verification time. Taking $\lambda = d$, the verification time satisfies the following transition (below we use notation $O_{\lambda,S}$ to hide $\mathrm{poly}(\lambda, S)$ terms):

- $\mathsf{Vtime}_{1,k} = O_{\lambda,S}(1)$;
- $\mathsf{Vtime}_{T,1} = \mathsf{Vtime}_{T/\lambda,\lambda} + O_{\lambda,S}(1)$;
- $\mathsf{Vtime}_{T,k} = \mathsf{Vtime}_{T/\lambda,k\lambda} + \mathsf{Vtime}_{T,k/\lambda} + O_{\lambda,S}(1)$.

Standard calculation shows that $\mathsf{Vtime}_{T,1}$ equals $O_{\lambda,S}(C_{\log_\lambda T})$, where $C_\ell \approx 2^{2\ell}/\mathrm{poly}(\ell)$ is the ℓ-th Catalan Number. Taking $\lambda = 2^{\sqrt{\log T}}$ gives us the desired $\mathsf{Vtime}_{T,1} = O_{\lambda,S}(2^{\sqrt{\log T}})$, and in particular $\mathsf{Vtime}_{T,1} = \mathrm{poly}(n)$ when $T = n^{O(\log n)}$.

3 Preliminaries

We adhere to the conventions in [BGHK25], hence many definitions and lemmas in this section are taken verbatim from that work.

- Lower case letters a, b mean scalars, while bolded lower case letters $\boldsymbol{a}, \boldsymbol{b}$ mean vectors. Upper case bold letters like $\boldsymbol{A}, \boldsymbol{B}$ are matrices.
- For an integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. When it is clear from context, we use $[0, n]$ to denote the set $\{0, 1, \ldots, n\}$. We let $\mathbb{F}$ denote a finite field. $\mathbb{GF}(2)$ denotes the finite field with 2 elements.
- For a vector $\boldsymbol{u} \in \mathbb{F}^n$, and a subset $\mathcal{S} \subset [n]$, $\boldsymbol{u}|_{\mathcal{S}} \in \mathbb{F}^{|\mathcal{S}|}$ denotes the subvector of $\boldsymbol{u}$ indexed by $\mathcal{S}$. For a sequence of vectors $\{\boldsymbol{u}_i\}_{i\in\mathcal{I}}$, where each $\boldsymbol{u}_i \in \mathbb{F}^n$, the notation $(\boldsymbol{u}_i)_{i\in\mathcal{I}} \in \mathbb{F}^{n\times|\mathcal{I}|}$ represents the matrix whose columns are the vectors $\boldsymbol{u}_i$ for every $i \in \mathcal{I}$.
- Given a matrix $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n) \in \mathbb{F}^{m\times n}$, $\boldsymbol{a}_j \in \mathbb{F}^m$ denotes its j-th column. We also use $\boldsymbol{A}[i, :]$ to denote the i-th row of A. We use $\boldsymbol{A}[:, 0 : j]$ or $\boldsymbol{A}[:, : j]$ to denote the submatrix of $\boldsymbol{A}$ consisting of the first j columns, and use $\boldsymbol{A}[:, -j :]$ to denote the submatrix of $\boldsymbol{A}$ consisting of the last j columns. For a subset $\mathcal{S} \subset [m]$, $\boldsymbol{A}[\mathcal{S}, :]$ denotes the submatrix of $\boldsymbol{A}$ consisting of rows indexed by $\mathcal{S}$.
- $\Delta(\boldsymbol{u}, \boldsymbol{v})$ is the (absolute) Hamming distance between the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Denote by $\Delta(\boldsymbol{u})$ the vector $\boldsymbol{u}$'s *Hamming weight*, defined to be the number of non-zero elements in $\boldsymbol{u}$. For any $d \in \mathbb{N}$, two vectors $\boldsymbol{u}, \boldsymbol{v}$ are *d-close* (in Hamming distance) if $\Delta(\boldsymbol{u}, \boldsymbol{v}) \le d$, and *d-far* otherwise. On a linear space $\mathcal{U} \subset \mathbb{F}^a$, let $\Delta(\mathcal{U}) \coloneqq \min_{\boldsymbol{u}\in\mathcal{U},\boldsymbol{u}\neq\boldsymbol{0}} \Delta(\boldsymbol{u})$.
- Given a Boolean circuit V, $\mathsf{size}(\mathsf{V})$ is the number of gates in the circuit, and $\mathsf{depth}(\mathsf{V})$ is the depth of the circuit.
- Given a Turing machine $\mathcal{M}$, we use $\langle\mathcal{M}\rangle$ to denote its constant-size description. Denote by $\mathsf{TISP}(T, S)$ the class of languages decidable by a Turing machine in time $T(n)$ and space $S(n)$.

3.1 Finite Fields and Distances on Matrices

All finite fields $\mathbb{F}$ considered in this work are always *constructible* in the following sense.

Definition 1 (Constructible Field Ensemble)**.** We say a field ensemble $\mathbb{F} = (\mathbb{F}_n)_{n\in\mathbb{N}}$ is *constructible* if every element in $\mathbb{F}_n$ has an $O(\log|\mathbb{F}_n|)$-bit representation, and addition, multiplication, and inverses can be computed in $\mathrm{polylog}(|\mathbb{F}_n|)$ time given the representations.

It is well known that for every $S = S(n)$, constructible field ensembles $\mathbb{F} = (\mathbb{F}_n)$ with $|\mathbb{F}_n| = \Theta(S)$ exist. Moreover, we make the (mild) assumption that addition, multiplication and inverses can be computed in $\widetilde{O}(\log|\mathbb{F}_n|)$ time, where $\widetilde{O}$ omits $\mathrm{polylog}\log(|\mathbb{F}_n|)$ factors. This implies that degree-d polynomials in $\mathbb{F}[X]$ can be evaluated in $\widetilde{O}(d\log|\mathbb{F}_n|)$ time (with $\widetilde{O}$ omitting $\mathrm{polylog}(d, \log|\mathbb{F}_n|)$ factors). These properties are satisfied in fields that support FFT. (See table 8.6 in [vzGG13])

An important metric defined on matrices, denoted by Δ_c, is as follows.

Definition 2 (Δ_c-distance)**.** Let $\mathbb{F}$ be a finite field. For matrices $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_L) \in \mathbb{F}^{k\times L}$ and $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_L) \in \mathbb{F}^{k\times L}$, the Δ_c-*distance* between $\boldsymbol{A}$ and $\boldsymbol{B}$, denoted by $\Delta_c(\boldsymbol{A}, \boldsymbol{B})$, is the maximum Hamming distance between corresponding columns of $\boldsymbol{A}$ and $\boldsymbol{B}$, i.e.,

$$\Delta_c(\boldsymbol{A}, \boldsymbol{B}) = \max_{i\in[L]} \Delta(\boldsymbol{a}_i, \boldsymbol{b}_i).$$

If $\Delta_c(\boldsymbol{A}, \boldsymbol{B}) \le d$, we say $\boldsymbol{A}$ and $\boldsymbol{B}$ are Δ_c-d-*close*. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are not Δ_c-d-close, then we say they are Δ_c-d-*far*.

Furthermore, given a distance parameter $d \in \mathbb{N}$ and $\boldsymbol{A} \in \mathbb{F}^{k\times L}$, define Δ_c-d-*ball* as

$$\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{A}) \coloneqq \left\{\boldsymbol{A}' \in \mathbb{F}^{k\times L} : \Delta_c(\boldsymbol{A}, \boldsymbol{A}') \le d\right\}.$$

Remark 2. Observe that Δ_c is a lower bound on how many rows have to be modified to transform one matrix into another.

3.2 Succinct Descriptions of Sets and Functions

Definition 3 (Uniform Arithmetic Circuits)**.** Let $\mathbb{F}$ be a field. Let $C = \{C_n : \mathbb{F}^n \to \mathbb{F}^m\}_{n\in\mathbb{N}}$ be a family of arithmetic circuits, consisting of fan-in 2 ADD and MULT gates over a field $\mathbb{F}$. For any $f = f(n)$, we say that C is f-space uniform if there exists a fixed $O(f(n))$-space Turing machine $\mathcal{M}$ that, on input 1^n, outputs the full description of the circuit C_n. When n is clear from the context, we omit the subscript n and write C instead of C_n.

An important special case is when $f(n) = \log(n)$, in which case we say that C is log-space uniform.

Remark 3. Any f-space uniform Boolean circuit C can be trivially extended to a f-space uniform arithmetic circuit C' of the same size and depth over any field $\mathbb{F}$.

The following *succinct descriptions* of sets can be used to recover the entire set.

Definition 4 (Descriptions of Sets)**.** A bit string $\langle\mathcal{S}\rangle \in \{0,1\}^B$ is a description of a set $\mathcal{S} = \{s_1, \ldots, s_k\} \subset \{0,1\}^p$ if there exists a (multi-output) p-space uniform circuit $G : [k] \times \{0,1\}^B \to \{0,1\}^p$ of fan-in 2, called its *implementation circuit*, such that $G(i, \langle\mathcal{S}\rangle) = s_i$ for all $i \in [k]$. The description is succinct if $|\langle\mathcal{S}\rangle| = B < k \cdot p$.

Similarly, *succinct descriptions* of functions can be used to configure a uniform circuit family to implement the function.

Definition 5 (Descriptions of Functions)**.** Let $\mathbb{F}$ be a field. We say that $\langle\Phi\rangle \in \{0,1\}^B$ is a description of a function $\Phi : \mathbb{F}^n \to \mathbb{F}^m$ if there exists a log-space uniform circuit $C : \mathbb{F}^{n+B} \to \mathbb{F}^m$ of fan-in 2, called its *implementation circuit*, such that $C(x, \langle\Phi\rangle) = \Phi(x)$ for all $x \in \mathbb{F}^n$. The description is succinct if $|\langle\Phi\rangle| = B < \mathsf{size}(\Phi)$.

When we write $\langle\Phi\rangle = \top$, we mean that Φ is the trivial predicate that outputs 1 on all inputs (i.e., it imposes no constraint on x).

The Turing machines that generate the uniform G and C in Definitions 4 and 5 take in their "shape parameters" $(k, 1^B, 1^p)$ and $(1^n, 1^B)$ as input, respectively.

3.3 Unique-Decoding Checksums

Let $\rho \in \mathbb{N}$ be a *deviation radius* and $R_{\mathsf{cksum}} \in \mathbb{N}$ be a checksum-length parameter. We use syndromes of linear error-correcting codes as checksums.

Definition 6 (Unique-Decoding Checksums)**.** Let $k, \rho, R_{\mathsf{cksum}} \in \mathbb{N}$ and let $\mathbb{F}$ be a field. A function $\mathsf{cksum}_\rho : \mathbb{F}^k \to \mathbb{F}^{R_{\mathsf{cksum}}}$ is a *ρ-unique decoding checksum function* if for any $\boldsymbol{m} \in \mathbb{F}^k$, and for any $\boldsymbol{m}', \boldsymbol{m}'' \in \mathbb{F}^k$ that are both ρ-close to $\boldsymbol{m}$ and $\boldsymbol{m}' \neq \boldsymbol{m}''$, $\mathsf{cksum}_\rho(\boldsymbol{m}') \neq \mathsf{cksum}_\rho(\boldsymbol{m}'')$.

The term *unique decoding* refers to the fact that if we know $\boldsymbol{m}'$ is ρ-close to some (fixed) $\boldsymbol{m}$, then we can uniquely determine $\boldsymbol{m}'$ given $\mathsf{cksum}_\rho(\boldsymbol{m}')$.

Proposition 1 (Linear-code checksums)**.** *Let $\mathcal{C} \subseteq \mathbb{F}^k$ be a linear code of minimum distance D and codimension R_{cksum}, and let $H \in \mathbb{F}^{R_{\mathsf{cksum}} \times k}$ be a parity-check matrix for $\mathcal{C}$. For any $\rho \in \mathbb{N}$ satisfying $2\rho < D$, the syndrome map*

$$\mathsf{cksum}_\rho(\boldsymbol{m}) \coloneqq H\boldsymbol{m}$$

is a ρ-unique decoding checksum function. Moreover, with $\widetilde{O}$ hiding polylogarithmic factors, cksum_ρ can be evaluated in $\widetilde{O}(kR_{\mathsf{cksum}}(\mathrm{polylog}|\mathbb{F}|))$, and its implementation circuit over $\mathbb{F}$ has size $O(kR_{\mathsf{cksum}})$ and depth $\widetilde{O}(1)$.

Proof. Fix $\boldsymbol{m} \in \mathbb{F}^k$ and let $\boldsymbol{m}', \boldsymbol{m}'' \in \mathbb{F}^k$ be distinct vectors that are both ρ-close to $\boldsymbol{m}$. Then $\boldsymbol{z} = \boldsymbol{m}' - \boldsymbol{m}''$ is nonzero and has Hamming weight at most 2ρ. If $\mathsf{cksum}_\rho(\boldsymbol{m}') = \mathsf{cksum}_\rho(\boldsymbol{m}'')$, then $H\boldsymbol{z} = H\boldsymbol{m}' - H\boldsymbol{m}'' = \boldsymbol{0}$ by linearity, so $\boldsymbol{z} \in \ker(H) = \mathcal{C}$. This is a nonzero codeword of Hamming weight at most $2\rho < D$, contradicting the minimum distance of $\mathcal{C}$. Hence $\mathsf{cksum}_\rho(\boldsymbol{m}') \neq \mathsf{cksum}_\rho(\boldsymbol{m}'')$. □

Lemma 1 (Reed–Solomon checksums)**.** *Let $k, d \in \mathbb{N}$ and let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq k$. Set $R_{\mathsf{cksum}} \coloneqq 2d$. Then a d-unique decoding checksum function $\mathsf{cksum}_d : \mathbb{F}^k \to \mathbb{F}^{R_{\mathsf{cksum}}}$ exists. With $\widetilde{O}$ hiding $\mathrm{polylog}(k)$ factors, $\mathsf{cksum}_d : \mathbb{F}^k \to \mathbb{F}^{R_{\mathsf{cksum}}}$ can be evaluated in $\widetilde{O}(kR_{\mathsf{cksum}}\mathrm{polylog}|\mathbb{F}|)$ time, and its implementation circuit over $\mathbb{F}$ has size $O(kR_{\mathsf{cksum}})$ and depth $\widetilde{O}(1)$.*

Proof. If $2d < k$, let $H \in \mathbb{F}^{2d \times k}$ be a parity-check matrix of a Reed–Solomon code over $\mathbb{F}$ with block length k, dimension $k - 2d$, and minimum distance $2d + 1$. Define $\mathsf{cksum}_d(\boldsymbol{m}) \coloneqq H\boldsymbol{m}$. By the previous proposition, this is a d-unique decoding checksum function.

If $2d \geq k$, let cksum_d be the identity map on $\mathbb{F}^k$ padded with $2d - k$ zero coordinates. Then cksum_d is injective, so it is d-unique decoding.

The evaluation bound follows by computing the R_{cksum} linear forms defining cksum_d. □

Generalizing the notion of checksums to matrices, given a matrix $\boldsymbol{M} \in \mathbb{F}^{k \times L}$ with columns $(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_L)$, we use $\mathsf{cksum}_d(\boldsymbol{M})$ to denote $\chi = (\mathsf{cksum}_d(\boldsymbol{m}_1), \ldots, \mathsf{cksum}_d(\boldsymbol{m}_L)) \in \mathbb{F}^{R_{\mathsf{cksum}} \times L}$.

3.4 Interactive Protocols

An $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime}, \Sigma)$-protocol is a *public-coin, ℓ-round interactive protocol, with alphabet Σ, per-round message length a, and prover runtime* Ptime *and verifier runtime* Vtime. Specifically, such a protocol consists of a pair of interacting Turing machines $(\mathsf{P}(x), \mathsf{V}(y))_{[\mathsf{pp}]}$, where the P has input a string $x \in \{0,1\}^*$, and the V has input a string $y \in \{0,1\}^*$, and both parties have explicit access to the input parameters pp. We omit x, y, and pp from the notation when they are not important to the context. The machines may also take in other private parameters as additional input (in particular, the prover might have some extra information that makes it more efficient), but we omit them from the notation for simplicity. The machine P is deterministic and runs in time Ptime, and is called *the prover*, while V is probabilistic, runs in time Vtime, and is called *the verifier*. P and V are the *two parties* of the protocol. We omit the specification of Σ when $\Sigma = \{0,1\}$.

In each round $j \in [\ell]$ of the protocol:

1. V sends a random message $q_j \leftarrow_R \Sigma^a$ to P, referred to as a *query*.
2. P responds with a message $a_j \in \Sigma^a$ determined by the prescribed next-message function, referred to as an *answer*, which (abusing notation) is denoted by

$$a_j \coloneqq \mathsf{P}(x, j, (q_1, \ldots, q_j)) \in \Sigma^a.$$

We make the simplifying assumption that both parties send messages of equal length, and that V never rejects in the middle of an execution. We denote the sequence of verifier random coins by $\boldsymbol{q} \coloneqq (q_1, \ldots, q_\ell)$. Finally, $\mathsf{P}(x, \boldsymbol{q}) \coloneqq (a_1, \ldots, a_\ell)$, where the $a_j = \mathsf{P}(x, j, \boldsymbol{q}_{\leq j})$ are the prescribed messages, and the notation $\langle \mathsf{P}(x), \mathsf{V}(y; \boldsymbol{q}) \rangle_{[\mathsf{pp}]} \in \{0,1\}^* \cup \{\perp\}$ represents the output of the verifier at the end of the interaction, which contains a special symbol $\perp$ to indicate that the verifier rejected. Note that the randomness is only over the random coins $\boldsymbol{q}$ of V, and we omit $\boldsymbol{q}$ and pp from the notation when they are not important in the context.

The *total communication complexity* of the protocol is the number of bits exchanged between the prover and the verifier, i.e. $2a\ell \cdot \log(|\Sigma|)$.

Let $\mathcal{L} \subset \{0,1\}^*$ be a language. We next define the notion of an interactive proof of $\mathcal{L}$.

Definition 7 (ϵ-Sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-$\mathsf{IP}$)**.** An $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-protocol $(\mathsf{P}(x), \mathsf{V}(y))_{[\mathsf{pp}]}$, where the public parameter is pp and $y = x$, is an *interactive proof* (IP) with soundness error ϵ for a language $\mathcal{L}$, if it satisfies the following completeness and soundness conditions.

- **Completeness:** For any $x \in \mathcal{L}$, there exists a prover strategy P such that

$$\Pr[\langle \mathsf{P}(x), \mathsf{V}(x) \rangle_{\mathsf{pp}} \neq \perp] = 1.$$

- ϵ**-Soundness:** For any $x \notin \mathcal{L}$ and any (computationally unbounded) prover strategy P^*,

$$\Pr[\langle \mathsf{P}^*(x), \mathsf{V}(x) \rangle_{\mathsf{pp}} \neq \perp] \leq \epsilon.$$

3.4.1 Low-depth-predicate Interactive Proof and the GKR protocol

We consider the following special type of protocol where the verifier does not query the input x but instead outputs a low-depth predicate to be checked against the input.

Definition 8 (ϵ-Sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-LDP-IP)**.** Let $\mathbb{F}$ be a field. An *low-depth-predicate-interactive-proof* (LDP-IP) with respect to a field $\mathbb{F}$ for a language $\mathcal{L}$ is a protocol with public parameters pp, where the prover is given some input $x \in \{0,1\}^n$, and the verifier receives no input (except the protocol's parameters). At the end of the protocol, the verifier either rejects (outputs $\perp$) or outputs the description of a low-depth predicate Ψ, which takes as input x and outputs 0 or 1, satisfying the following:

- **Completeness:** If $x \in \mathcal{L}$, there exists a prover strategy P such that

$$\Pr[\langle \mathsf{P}(x), \mathsf{V} \rangle_{[\mathsf{pp}]} = \langle \Psi \rangle \text{ s.t. } \Psi(x) = 1] = 1,$$

- **Soundness:** If $x \notin \mathcal{L}$, then for any (computationally unbounded) prover strategy P^*,

$$\Pr[\langle \mathsf{P}^*(x), \mathsf{V} \rangle_{[\mathsf{pp}]} = \langle \Psi \rangle \text{ s.t. } \Psi(x) = 1] \leq \epsilon.$$

Importantly, the verifier never accesses x throughout the protocol.

We restate the main result from [GKR15] in terms of LDP-IP.

Theorem 3 (The GKR Protocol, as an LDP-IP)**.** *Let $\mathbb{F}$ be a field, and let $\Phi : \mathbb{F}^n \to \mathbb{F}$ be an arithmetic circuit with addition and multiplication gates of fan-in 2 over $\mathbb{F}$, with description $\langle \Phi \rangle$. Let $G(x, \langle \Phi \rangle)$ be the log-space uniform circuit that outputs $\Phi(x)$ on input $x \in \mathbb{F}^n$ and $\langle \Phi \rangle \in \{0,1\}^{|\langle \Phi \rangle|}$. Denote the depth and size of G by $D = D(n) \geq \log n$ and $S = S(n) \geq n$.*

For some constant $C_{\mathsf{GKR}} > 0$, there exists an ϵ-sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-LDP-IP, abbreviated as $\mathsf{GKR} \coloneqq (\mathsf{P}(x), \mathsf{V})_{[\mathbb{F}, \langle \Phi \rangle]}$, for the language $\mathcal{L}_\Phi = \{x \in \mathbb{F}^n : \Phi(x) = 1\}$, with the following complexity (with $\widetilde{O}$ ignoring poly-logarithmic factors in $D, \log S$):

- *The soundness error is bounded by $\epsilon \leq C_{\mathsf{GKR}} \cdot \left(\frac{D \log S}{|\mathbb{F}|}\right) = O(\frac{D \log S}{|\mathbb{F}|})$.*
- *$\ell = O(D \cdot \log S)$.*
- *$a = O(\log |\mathbb{F}|)$.*
- *$\mathsf{Ptime} = \widetilde{O}(\mathrm{poly}(S) \cdot \mathrm{polylog}|\mathbb{F}|)$.*
- *$\mathsf{Vtime} = \widetilde{O}(D \log S \cdot \log |\mathbb{F}| + |\langle \Phi \rangle| \cdot \log |\mathbb{F}|)$ (and V does not access x).*

$|\langle \Psi \rangle| = O(\log |\mathbb{F}|)$, and the implementation circuit C for Ψ satisfies

- *$\mathsf{size}(C) = \widetilde{O}(n)$.*
- *$\mathsf{depth}(C) = \widetilde{O}(1)$.*

3.4.2 Interactive Proof of Proximity with Row Reduction

An Interactive Proof of Proximity with Row Reduction allows us to reduce checking a predicate Φ over a matrix $\boldsymbol{M}$ to checking a related predicate Ψ over a subset of rows of $\boldsymbol{M}$. It is a generalization of the standard IPP, which is defined over bit strings and with respect to Hamming distance.

Definition 9 (ϵ-Sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-$\mathsf{IPP}$ with Row Reduction)**.** Let $\mathbb{F}$ be a field, $k, L \in \mathbb{N}$, and $\Phi : \mathbb{F}^{k\times L} \to \{0,1\}$ be a predicate. An ϵ-sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$ *Interactive Proof of Proximity* (IPP) with Row Reduction for the language

$$\mathcal{L}_\Phi \coloneqq \left\{\boldsymbol{M} \in \mathbb{F}^{k\times L} : \Phi(\boldsymbol{M}) = 1\right\},$$

is an $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-protocol, abbreviated as $\mathsf{IPP} \coloneqq (\mathsf{P}(\boldsymbol{M}), \mathsf{V})_{[\mathbb{F},\langle\Phi\rangle,d]}$ whose prover input is a matrix $\boldsymbol{M} \in \mathbb{F}^{k\times L}$, and the verifier receives no input (except the protocol's parameters). At the end of the protocol, the verifier outputs either $\perp$ or $(\langle\mathcal{Q}\rangle, \langle\Psi\rangle)$, where $\mathcal{Q} \subsetneq [k]$ is a set of rows and Ψ is a predicate such that the following holds.

- **Completeness:** If $\Phi(\boldsymbol{M}) = 1$, then $\Pr[\Psi(\boldsymbol{M}[\mathcal{Q}, :]) = 1] = 1$.
- ϵ**-Soundness:** Suppose $\boldsymbol{M}$ is d-Δ_c-far from $\mathcal{L}_\Phi$, *i.e.* $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) \cap \mathcal{L}_\Phi = \varnothing$,[2] then for any prover strategy P^*,

$$\Pr\left[\langle \mathsf{P}^*(\boldsymbol{M}), \mathsf{V}\rangle_{[\mathbb{F},\langle\Phi\rangle,d]} = (\langle\mathcal{Q}\rangle, \langle\Psi\rangle) \text{ s.t. } \Psi(\boldsymbol{M}[\mathcal{Q}, :]) = 1\right] \le \epsilon.$$

For the IPP to be non-trivial, we require $|\mathcal{Q}| \ll k$. Such an IPP exists by the following theorem, which generalizes Lemma 4 in [BGHK25].

Theorem 4. *There exists a constant $c > 0$ such that for all σ, d, k, $L \in \mathbb{N}$, and a field $\mathbb{F}$, and any description $\langle\Phi\rangle$ of a predicate Φ with implementation circuit C whose size and depth are S and D, respectively, if the following holds:*

- $d = \Omega(\sigma \log k)$,
- $|\mathbb{F}| = \Omega(2^\sigma \cdot ((\sigma dL \log(|\mathbb{F}|k))^c + D \log S))$,

then there exists a $2^{-\sigma}$-sound IPP with row reduction, abbreviated by $\mathsf{IPP}(\Phi) \coloneqq (\mathsf{P}(\boldsymbol{M}), \mathsf{V})_{[\sigma,\mathbb{F},\langle\Phi\rangle,d]}$, for the input matrix $\boldsymbol{M} \in \{0,1\}^{k\times L}$, and its output subset $\mathcal{Q} \subset [k]$ satisfies:

$$|\mathcal{Q}| \le \left\lceil 24\sigma \cdot \frac{k}{d} \right\rceil.$$

Let $\widetilde{O}$ omit $\mathrm{polylog}(|\mathbb{F}|, k, L)$ *factors. The complexity of the protocol is as follows.*

- $\ell = \widetilde{O}(D \log S)$.
- $a = \widetilde{O}(dL + \mathrm{poly}(d))$.
- $\mathsf{Ptime} = \mathrm{poly}(k, L, d, S, \log|\mathbb{F}|)$.

[2] Recall that $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M})$ is the set of all matrices that are Δ_c-d-close to $\boldsymbol{M}$

- $\mathsf{Vtime} = \widetilde{O}(dL(D \log S + |\langle \Phi \rangle|) + \mathrm{poly}(d))$.

$|\langle \mathcal{Q} \rangle| = \widetilde{O}(\mathrm{poly}(d))$ *and* $|\langle \Psi \rangle| = \widetilde{O}(L + \mathrm{poly}(d))$. *Let* G *and* C *be the implementation circuits of* $\mathcal{Q}$ *and* Ψ *respectively. They satisfy the following.*

- $\mathsf{size}(G) = \widetilde{O}(\mathrm{poly}(d))$.
- $\mathsf{depth}(G) = \widetilde{O}(1)$.
- $\mathsf{size}(C) = \widetilde{O}(|\mathcal{Q}| \cdot L)$.
- $\mathsf{depth}(C) = \widetilde{O}(1)$.

The specific requirements on d *and* $|\mathbb{F}|$ *are*

- $d \geq 48\sigma \log k$,
- $|\mathbb{F}| \geq C_{\mathsf{GKR}} \cdot 2^{\sigma+4c+7}((\sigma dL \log(|\mathbb{F}|(k+1)))^c + \mathsf{depth}(C) \log \mathsf{size}(C) + 1)$,

where C_{GKR} *is the constant in Theorem 3.*

The construction utilizes the GKR protocol (Theorem 3) as well as a special IPP for the *polynomial valuation language* (PVAL) with row reduction (Theorem 9 in [BGHK25]; c.f. [RR20]). For completeness, we provide the corresponding definitions and the proof of Theorem 4 in Section A.

4 Doubly-Efficient Proof for Space-bounded Computation

We state our main result as follows.

Theorem 5 (Formal Statement of Theorem 2)**.** *For all* $n, T = T(n) > n, S = S(n), \sigma = \sigma(n) \in \mathbb{N}$*, there exists a* $2^{-\sigma}$*-sound* $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$*-IP for deciding any language in* $\mathsf{TISP}(T, S)$*. The complexity of the protocol is as follows.*

- $\ell = 2^{O(\sqrt{\log T})} \cdot \mathrm{poly}(\sigma, \log S)$.
- $a = 2^{O(\sqrt{\log T})} \cdot S \cdot \mathrm{poly}(\sigma, \log S)$.
- $\mathsf{Ptime} = \mathrm{poly}(S, T, \sigma)$.
- $\mathsf{Vtime} = 2^{O(\sqrt{\log T})} \cdot S^2 \cdot \mathrm{poly}(\sigma, \log S)$.

Letting $T = n^{O(\log n)}$ in the above theorem, we obtain a doubly efficient interactive proof system for every PSPACE language decidable in time $n^{O(\log n)}$.

Corollary 1 (Formal Statement of Theorem 1)**.** *For all* $n, T = T(n)$ *such that* $n < T < n^{O(\log n)}$*,* $S = S(n) \in \mathrm{poly}(n), \sigma = \sigma(n) \in \mathbb{N}$*, there exists a* $2^{-\sigma}$*-sound* $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$*-IP for deciding any language in* $\mathsf{TISP}(T, S)$*. The complexity of the protocol is as follows.*

- $\ell = \mathrm{poly}(n, \sigma)$.
- $a = S \cdot \mathrm{poly}(n, \sigma)$.
- $\mathsf{Ptime} = \mathrm{poly}(S, T, \sigma)$.
- $\mathsf{Vtime} = S^2 \cdot \mathrm{poly}(n, \sigma)$.

4.1 LDP-IP for the Batch Language

We prove our main theorems by constructing a *Low-Depth-Predicate*-IP, or LDP-IP (as defined in Section 3.4.1, Definition 8), for verifying a batch of claims for deterministic time-t computations, which we call transition claims. Recall that an LDP-IP is a special protocol where the verifier never accesses the input throughout the protocol execution, and only outputs a description of a low-depth predicate Ψ about the input $\boldsymbol{x}$. We note that our approach deviates from prior works [RRR16, BGHK25], which constructed doubly efficient interactive proofs by first constructing protocols for verifying a batch of unambiguous interactive proofs.

Formally, fix a Turing machine $\mathcal{M}$ with time complexity $T(n)$ and space complexity $S(n)$. For any $t \leq T(n)$, let the language $\mathcal{L}_t$ consist of all pairs $(x_{\mathsf{start}}, x_{\mathsf{end}}) \in \{0,1\}^{S(n)} \times \{0,1\}^{S(n)}$ such that $\mathcal{M}$ transitions from configuration x_{start} to configuration x_{end} in exactly t steps. Given a batch size parameter $k = k(n) \in \mathbb{N}$, define the batch language

$$\mathcal{L}_t^k := \{((x_{i,\mathsf{start}}, x_{i,\mathsf{end}}))_{i\in[k]} \mid \forall i \in [k]\ (x_{i,\mathsf{start}}, x_{i,\mathsf{end}}) \in \mathcal{L}_t\}.$$

Theorem 6. *For all $n \in \mathbb{N}$, $S = S(n), T = T(n)$, $t = t(n) \leq T, k = k(n), \sigma = \sigma(n) \in \mathbb{N}$, there exists an upper bound*

$$\Lambda(T,k) \in 2^{O\left(\sqrt{\log \binom{2\log T + \log k}{\log T}}\right)},$$

and an ϵ-sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-LDP-IP, denoted as $\mathsf{Batch}(t,k)$, for the batch language $\mathcal{L}_t^k$, where

- $\epsilon = 2^{-\sigma}$.
- $\ell = \Lambda(T,k) \cdot \mathrm{poly}(\sigma)$.
- $a = \Lambda(T,k) \cdot S \cdot \mathrm{poly}(\sigma)$.
- $\mathsf{Ptime} = \mathrm{poly}(T,k,S,\sigma)$,
- $\mathsf{Vtime} = \Lambda(T,k) \cdot S^2 \cdot \mathrm{poly}(\sigma)$.

Furthermore, the output $\langle\Psi\rangle$ defines a low-depth predicate Ψ on $\boldsymbol{x}$, whose implementation circuit C_Ψ satisfies

- $\mathsf{size}(C_\Psi) = (k + \Lambda(T,k)) \cdot S \cdot \mathrm{poly}(\sigma)$.
- $\mathsf{depth}(C_\Psi) = \Lambda(T,k) \cdot \mathrm{poly}(\sigma)$.

Finally, the description length $|\langle\Psi\rangle| = O(\Lambda(T,k) \cdot \mathrm{poly}(\sigma) \cdot S)$.

With Theorem 6, our proof of Theorem 5 is straightforward.

Proof of Theorem 5. This follows from Theorem 6.

1. Let x be the input to $\mathcal{M}$. P first simulates $\mathcal{M}(x)$ for T steps and stores its tableau $\boldsymbol{\tau} \in \{0,1\}^{T\times S}$ to be used as auxiliary information to the sub-protocols.

2. Both parties select the appropriate parameters, then run the protocol given by Theorem 6, denoted as $\mathsf{Batch}(T, 1)$, which outputs a predicate description $\langle \Psi \rangle$ to the verifier.

3. V lets $\boldsymbol{x} = (x_{\mathsf{start}}, x_{\mathsf{end}})$, where x_{start} denotes the start state with x as input, and x_{end} denotes the accept state of $\mathcal{M}$.

4. V accepts iff $C_\Psi(\boldsymbol{x}, \langle \Psi \rangle) = 1$.

An LDP-IP implies a standard IP when the verifier V has access to the input $\boldsymbol{x}$ in the clear.

Plugging in $k = 1$ into the bound $\Lambda(T, k)$, we have

$$\Lambda(T, 1) = 2^{O\left(\sqrt{\log \binom{2 \log T}{\log T}}\right)} = 2^{O(\sqrt{\log T})}.$$

The resulting IP has identical round-complexity ℓ and per-round communication complexity a as $\mathsf{Batch}(T, 1)$, and

- $\mathsf{Ptime}' = O(TS) + \mathsf{Ptime}_{\mathsf{LDP\text{-}IP}} = \mathrm{poly}(S, T, \sigma)$,
- $\mathsf{Vtime}' = O(\mathsf{size}(C_\Psi)) + \mathsf{Vtime}_{\mathsf{LDP\text{-}IP}} = 2^{O(\sqrt{\log T})} \cdot S^2 \cdot \mathrm{poly}(\sigma)$.

The additional terms are due to the prover's simulation of $\mathcal{M}$ and the verifier's evaluation of the circuit Ψ. □

5 Low-Depth-Predicate Interactive Proof $\mathsf{Batch}(t, k)$

In this section, we prove Theorem 6.

Road map of our construction Our protocol in Theorem 6 is constructed by recursion. Let $\lambda \in \mathbb{N}$ be some parameter. Recall that the parties try to verify $\boldsymbol{x} \in \mathcal{L}_t^k$. Let us denote the protocol for checking this as $\mathsf{Batch}(t, k)$. We shall try to reduce this to constructing some $\mathsf{Batch}(t', k')$ where either (1) $k' \ll k$ and $t' = t$, or (2) $k' = k$ and $t' \ll t$.[3] We denote these two reductions by R_1 and R_2, as illustrated in Figure 1. If both R_1 and R_2 can be performed doubly efficiently, then we are done, since we can just apply them sufficiently many times and reduce to the case where both k and t are small, and output the final low-depth predicate defined on only those instances.

When t is tiny, the GKR protocol applied to the circuit $\Phi(\boldsymbol{x})$, which verifies that all adjacent states are consistent with the transition rules of $\mathcal{M}$ (see Proposition 2), is of depth $\widetilde{O}(t)$, so we immediately obtain a doubly efficient LDP-IP.

The tricky case is when t is not tiny, as $\mathsf{depth}(\Phi) = \Omega(t)$, and the GKR verifier is no longer efficient. The good news is that we can reduce the number of instances k, by a careful application of an IPP with row reduction (Definition 9). Before applying this protocol, the verifier needs to prepare a claim Φ_{IPP} with Δ_c-distance to $\boldsymbol{x}$ when $\boldsymbol{x} \notin \mathcal{L}_t^k$. Naively, we can simply run $\mathsf{Batch}(t, k)$ to obtain the claim Φ_{IPP} — our LDP-IP is designed to output a false claim when $x \notin \mathcal{L}_t^k$! However, this is a catch-22 as we are exactly trying to construct $\mathsf{Batch}(t, k)$. Fortunately, there is another LDP-IP that we can use to obtain such a claim when $x \notin \mathcal{L}_t^k$, which will become clear when we consider the other boundary case — when k is tiny, hence let us consider this case.

[3] For (2), we ended up reducing to some (t', k') where $k' > k$ and $t' < t$, as in Figure 2. This is not ideal, but enough for the recursion to terminate because the two recursive calls still define a partial order on the grid of (t, k).

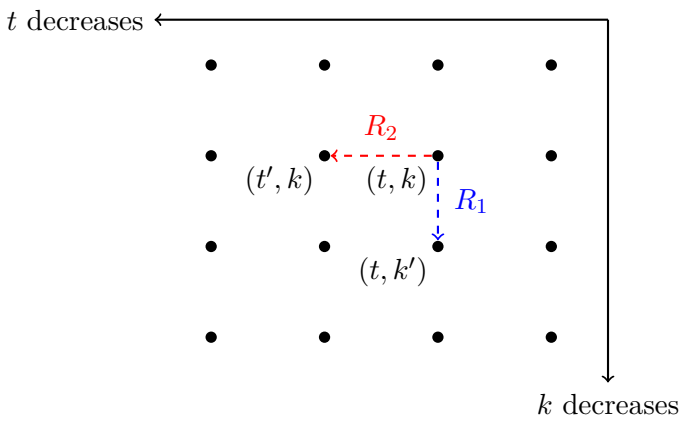

Figure 1: Grid of protocol dimensions $\mathsf{Batch}(t, k)$. Moving left decreases time t, and moving down decreases batch size k. The goal is to reduce to the case where both t and k are small.

When k is too tiny for instance reduction and t is still large, the prover can reduce t by λ by sending λ many equally spaced "midpoints" between the length-t path of x_{start} and x_{end} for each length-t transition statement $(x_{\mathsf{start}}, x_{\mathsf{end}})$, and both parties then need to apply the protocol $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$. Even though this *increases* k by a factor of λ, we are still making progress because in $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$, we can reduce the $(k \cdot \lambda)$-dimension by applying an IPP with row reduction again.

Going back to the case of general (t, k), we can apply a similar idea when we try to generate Δ_c distance. After putting the k statements on k rows, the prover introduces $\lambda - 1$ midpoint states for each statement, hence creating the midpoint matrix $\boldsymbol{M}$ (Definition 10). However, if the prover sends this matrix in the clear, prohibitively many — $\Omega(k \cdot \lambda)$ — states have to be sent. The good news is that the verifier can enforce a joint constraint Φ_{IPP} on the rows of $\boldsymbol{M}$, exactly because they can reduce to $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$ on $\boldsymbol{x}^{\nwarrow} \in \{0,1\}^{(k\cdot\lambda)\times(2S)}$, where each row in $\boldsymbol{x}^{\nwarrow}$ corresponds to a t/λ-length transition statement specified in $\boldsymbol{M}$. (Note that the symbol $\nwarrow$ refers to the input dimensions $(t/\lambda, k \cdot \lambda)$, which is situated in the upper-right corner of (t, k) in Figure 2.) Therefore, for some distance parameter $d = d(n)$, they proceed with the following two steps:

1. The prover only "commits" to these states by only sending the checksums χ of every column. These checksums are constructed from a distance-$\widetilde{O}(d)$ error correcting code, and force the prover to cheat on many rows if it cheats, as in previous literature.

2. Both parties then run $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$ on $\boldsymbol{x}^{\nwarrow}$.

The verifier then obtains a claim Φ_{IPP} that is false on $\boldsymbol{x}^{\nwarrow}$ (equivalently, $\boldsymbol{M}$) whenever $\boldsymbol{x}^{\nwarrow} \notin \mathcal{L}^{k\cdot\lambda}_{t/\lambda}$. With this additional constraint, the parties proceed with an IPP with row reduction for the language $\mathcal{L}_{\Phi_{\mathsf{IPP}}}$ on prover input $\boldsymbol{M}$. This protocol outputs a subset of roughly (k/d) rows, $\mathcal{Q} \subset [k]$, as well as a linear constraint Ψ_{IPP} over the sub-matrix $\boldsymbol{M}^{\downarrow} = \boldsymbol{M}[\mathcal{Q}, :]$. (Note that $\downarrow$ refers to $(t, k/d)$, which is situated beneath (t, k) in Figure 2.) Let $\boldsymbol{x}^{\downarrow}$ be the $|\mathcal{Q}|$ length-t statements specified by $\boldsymbol{M}^{\downarrow}$'s leftmost and rightmost columns. Lemma 4 guarantees that if the original $\boldsymbol{x} \notin \mathcal{L}^k_t$, then either $\boldsymbol{x}^{\downarrow} \notin \mathcal{L}^{k/d}_t$ or $\Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 0$. Owing to this additional linear constraint Ψ_{IPP}, which is defined on the reduced midpoint matrix $\boldsymbol{M}^{\downarrow}$, instead of just checking $\boldsymbol{x}^{\downarrow} \in \mathcal{L}^{k/d}_t$, we need to additionally

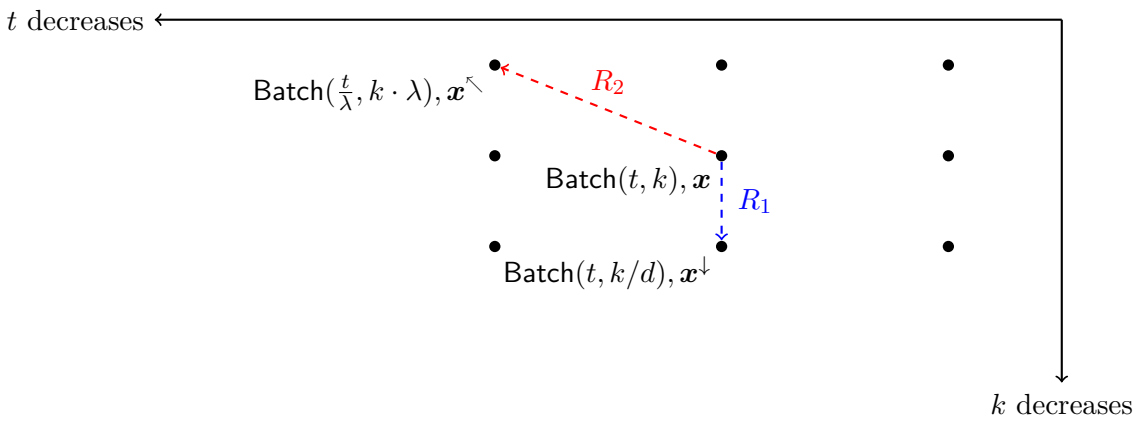

Figure 2: Grid of the sub-protocol dimensions that we actually recur to, and their respective batches of statements. The two arrows represent the two reduced protocols that we actually construct recursively.

check that the corresponding $\boldsymbol{M}^{\downarrow}$ satisfies $\Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 1$. This turns out to be straightforward by having $\mathsf{Batch}(t, k/d)$ perform this check on top of checking $\boldsymbol{x}^{\downarrow} \in \mathcal{L}_t^{k/d}$ throughout.

Finally, a minor technicality is that we have to ensure the resulting low-depth predicate is indeed defined over $\boldsymbol{x}$, and not just over a related string. This happens when we call Batch on dimensions other than (t, k), which we handle by redefining predicates over $\boldsymbol{x}$ as follows:

1. When k is large, $\mathsf{Batch}(t, k/d)$ returns a low-depth predicate over $\boldsymbol{x}^{\downarrow} \in \{0,1\}^{(k/d)\times(2S)}$.

   The actual output predicate over $\boldsymbol{x}$ first selects $\boldsymbol{x}^{\downarrow} = \boldsymbol{x}[\mathcal{Q}, :]$ from the input $\boldsymbol{x}$ and checks that $\boldsymbol{x}^{\downarrow}$ satisfies the low-depth predicate returned by $\mathsf{Batch}(t, k/d)$.

2. Conversely, when k is tiny, $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$ returns a low-depth predicate over the midpoint statements $\boldsymbol{x}^{\nwarrow} \in \{0,1\}^{(k\cdot\lambda)\times(2S)}$.

   The output predicate over $\boldsymbol{x}$ has the midpoint statements $\boldsymbol{x}^{\nwarrow}$ hard-coded, and verifies that $\boldsymbol{x}$ is consistent with $\boldsymbol{x}^{\nwarrow}$ on the boundaries and $\boldsymbol{x}^{\nwarrow}$ indeed satisfies the given low-depth predicate.

To sum up the case of general (t, k), our goal of constructing $\mathsf{Batch}(t, k)$ is reduced to constructing two sub-protocols (Figure 2):

(a) Calling $\mathsf{Batch}(t/\lambda, k \cdot \lambda)$ on $\boldsymbol{x}^{\nwarrow}$.

(b) Calling $\mathsf{Batch}(t, k/d)$ on $\boldsymbol{x}^{\downarrow}$.

In addition to these recursive calls, the other steps, including a call to the IPP with row reduction (Theorem 4), incur a fixed $\mathrm{poly}(\lambda, d, S)$ cost for the verifier. To optimize the efficiency guarantees, we set

$$\lambda := \left\lceil 2^{\sqrt{\log\binom{2\log T + \log k}{\log T}}} \right\rceil, d, k_{\mathsf{base}} \in \mathrm{poly}(\lambda).$$

Here we give some intuition for the efficiency bound for the case when $k = 1$ and $\lambda = 2^{\sqrt{\log T}}$. The full analysis is deferred to Section 5.2.3.

Let us focus on the running time of the verifier, which is given by the recurrence $\mathsf{Vtime}(t,k) = \mathsf{Vtime}(t/\lambda, k\cdot\lambda) + \mathsf{Vtime}(t, k/\lambda) + \mathrm{poly}(S,\lambda)$. For simplicity,[4] we derive a closed-form upper bound on $\mathsf{Vtime}(T,0)$ when $t_{\mathsf{base}} = k_{\mathsf{base}} = d = \lambda$ and $T = \lambda^\rho$. Write $t = \lambda^i$ and $k = \lambda^j$, and let $\mathsf{Vtime}'(i,j) = \mathsf{Vtime}(\lambda^i, \lambda^j)$. Then we can verify that Vtime' satisfies the following: if $i = 0$ (the base case), $\mathsf{Vtime}'(i,j) = \widetilde{O}(\lambda)$. Otherwise, if $j = 0$, $\mathsf{Vtime}'(i,j) = \mathsf{Vtime}'(i-1,j) + \mathrm{poly}(S,\lambda)$, and if $j \geq 1$, $\mathsf{Vtime}'(i,j) = \mathsf{Vtime}'(i-1,j+1) + \mathsf{Vtime}'(i,j-1) + \mathsf{poly}(S,\lambda)$. Consider the recursion tree for Vtime' rooted at node $(\rho,0)$. Our goal is to upper bound the number of nodes in this recursion tree, which would yield an upper bound on $\mathsf{Vtime}(T,0)$.

In this recursion tree, every node (i,j) that has more than one child has exactly two children, which correspond to two moves that we can make: a $\nwarrow$-move from (i,j) to $(i-1,j+1)$, and a $\downarrow$-move from (i,j) to $(i,j-1)$. Starting at node $(\rho,0)$, we must make exactly ρ $\nwarrow$-moves to reach a node $(0,j)$, i.e., the base case. Since after making ρ $\nwarrow$-moves we are at node $(0,\rho)$, and we can never reach a node (i,j) such that $j<0$, this implies that we can make at most ρ $\downarrow$-moves. Suppose at some step we have made exactly a $\nwarrow$-moves and b $\downarrow$-moves; then we are at node $(\rho-a, a-b)$, and since we never reach a node (i,j) such that $j<0$, this implies that a and b must satisfy $a-b\geq 0$. The number of paths starting from $(\rho,0)$ that make ρ $\nwarrow$-moves and ρ $\downarrow$-moves and at every step satisfy $a-b\geq 0$, is exactly the number of Dyck paths of size ρ, which is given by the ρth Catalan number $C_\rho = \binom{2\rho}{\rho}/(\rho+1)$. Since the number of nodes in the recursion tree is at most the number of such paths, we have that $\binom{2\rho}{\rho}/(\rho+1)\cdot\mathrm{poly}(\lambda,S) \leq 2^{2\rho}\cdot\mathrm{poly}(S)$ is an upper bound on $\mathsf{Vtime}(T,0)$. Since $\lambda = 2^{\sqrt{\log T}}$, $\rho = \log_\lambda T = \sqrt{\log T}$, and thus $\mathsf{Vtime}(T,0) = 2^{O(\sqrt{\log T})}\cdot\mathrm{poly}(S)$.

5.1 Our Construction

We reduce the number of transition claims k by iteratively applying the IPP with row reduction (Theorem 4). To ensure soundness when sub-sampling to a smaller batch of transitions, we augment the batch language with a predicate Φ. This predicate Φ enforces auxiliary constraints — such as consistency between the midpoints and the succinct checksums — that ensure the smaller batch remains false if the original batch was false. To implement this, we introduce the *midpoint matrix*.

Definition 10 (Midpoint Matrix)**.** Let $\boldsymbol{x} = ((x_{i,\mathsf{start}}, x_{i,\mathsf{end}}))_{i\in[k]}$ be an instance of $\mathcal{L}_t^k$ for some $t\geq\lambda$, where $\lambda\in\mathbb{N}$ is some splitting parameter. For all $i\in[k]$, $j\in\{0,\ldots,\lambda\}$, let $x_{i,j}$ denote the configuration that $\mathcal{M}$ reaches after $(\frac{t}{\lambda})\cdot j$ time steps when starting from configuration $x_{i,0} = x_{i,\mathsf{start}}$. Define the *midpoint matrix* $\boldsymbol{M} = \boldsymbol{M}_{\boldsymbol{x}} \in \{0,1\}^{k\times(S(\lambda+1))}$ as follows: for all $i\in[k]$, $j\in\{0,\ldots,\lambda\}$, $\boldsymbol{M}[i,j] = x_{i,j}$.

And the augmented language is

$$\mathcal{L}_t^k[\Phi] \coloneqq \{\boldsymbol{x}\in\{0,1\}^{k\times 2S} \mid \boldsymbol{x}\in\mathcal{L}_t^k \wedge (t\geq\lambda \implies \Phi(\boldsymbol{M}_{\boldsymbol{x}}) = 1)\}.$$

Initially, we set $\langle\Phi_0\rangle = \top$ to denote the fact that there is no additional constraint (and this predicate always outputs 1).

We show that $\mathcal{L}_t^k[\Phi]$ admits a doubly efficient $\mathsf{LDP\text{-}IP}$, which we call $\mathsf{Batch}(t,k)$.

[4] The actual set of parameters is different; see Protocol 1.

Theorem 7 (Protocol $\mathsf{Batch}(t,k)$, an LDP-IP for the augmented language)**.** *For $n \in \mathbb{N}$, let $S = S(n)$, $T = T(n) > n$, and $\sigma = \sigma(n)$. Suppose $t = t(n) \le T$ and $k = k(n)$. Let C be the implementation circuit for a predicate $\Phi : \{0,1\}^{k\times(S(\lambda+1))} \to \{0,1\}$.*

For some upper bound

$$\Lambda(T,k) \in 2^{O\left(\sqrt{\log\binom{2\log T+\log k}{\log T}}\right)},$$

there exists an ϵ-sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-LDP-IP (Protocol 1), abbreviated as $\mathsf{Batch}(t,k) \coloneqq (\mathsf{P}(\boldsymbol{x}), \mathsf{V})_{[S,T,t,k,\langle\Phi\rangle]}$ for the language $\mathcal{L}_t^k[\Phi]$. We additionally assume P has access to the entire tableau of the computation $\boldsymbol{\tau} \in \{0,1\}^{T\times S}$ as auxiliary input. The complexities of the protocol are as follows. Note that the bounds here are loose and use the fact that $t \le T$.

- $\epsilon \le 2^{-\sigma}$.
- $\ell = (\Lambda(T,k) + \mathsf{depth}(C)\log\mathsf{size}(C))\cdot\mathrm{poly}(\sigma)$.
- $a = \Lambda(T,k)\cdot S\cdot\mathrm{poly}(\sigma)$.
- $\mathsf{Ptime} = \mathrm{poly}(S,k,T,\sigma,\mathsf{size}(C))$.
- $\mathsf{Vtime} = (\Lambda(T,k) + \mathsf{depth}(C)\log\mathsf{size}(C) + |\langle\Phi\rangle|)\cdot S^2\cdot\mathrm{poly}(\sigma)$.

Furthermore, the output $\langle\Psi\rangle$ defines a low-depth predicate Ψ on $\boldsymbol{x}$, whose implementation circuit C_Ψ satisfies

- $\mathsf{size}(C_\Psi) = (k + \Lambda(T,k))\cdot\mathrm{poly}(\sigma)\cdot S$.
- $\mathsf{depth}(C_\Psi) = \Lambda(T,k)\cdot\mathrm{poly}(\sigma)$.

Finally, $|\langle\Psi\rangle| = \Lambda(T,k)\cdot\mathrm{poly}(\sigma)\cdot S$.

Theorem 6 follows from Theorem 7 by letting $\langle\Phi\rangle = \top$, defined to be the predicate that always outputs 1, in which case $\mathcal{L}_t^k[\Phi] = \mathcal{L}_t^k$ and the terms involving C and $|\langle\Phi\rangle|$ are all $O(1)$.

The protocol For a summary of notation, see Table 1. Let $\lambda = \mathrm{poly}(n)$ be a parameter. Without loss of generality, we assume $t = \lambda^\tau$ for some $\tau \in \mathbb{N}$. Let

$$N(T,k) \coloneqq \binom{2\log_\lambda T + \log_\lambda(k+1) + O(1)}{\log_\lambda T + O(1)}$$

denote the recurrence-tree node bound used in the soundness and complexity analyses, and set

$$\sigma_{\mathsf{loc}} \coloneqq \sigma + \lceil\log(4N(T,k))\rceil + 2.$$

For some absolute constants c_0, c_1 to be analyzed in Section 5.2.1, we let $|\mathbb{F}|$ be the smallest binary field such that $|\mathbb{F}| \ge c_0 2^{4\sigma_{\mathsf{loc}}} T^{10}(k+1)^4 S^{c_1}\cdot(\mathsf{depth}(C)\log\mathsf{size}(C)+1)$.

For the base case when $k < k_{\mathsf{base}}$, apply the following $\mathsf{SmallBatch}$ protocol.

Lemma 2 (Base case protocol for small batch size)**.** *For $n \in \mathbb{N}$, let $S = S(n)$, $T = T(n) > n$, and $\sigma = \sigma(n)$. Suppose $t = t(n) \le T$, $k < k_{\mathsf{base}}$, and $\lambda = \Lambda(T,k)$. Let C be the implementation circuit for the predicate $\Phi : \{0,1\}^{k\times(S(\lambda+1))} \to \{0,1\}$.*

There exists an ϵ-sound $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$-LDP-IP, abbreviated as $\mathsf{SmallBatch} \coloneqq (\mathsf{P}(\boldsymbol{x}), \mathsf{V})_{[S,T,t,k,\langle\Phi\rangle]}$ for the language $\mathcal{L}_t^k[\Phi]$. We additionally assume P has access to the entire tableau of the computation $\boldsymbol{\tau} \in \{0,1\}^{T\times S}$ as auxiliary input. The complexities of the protocol are identical to the ones stated in Theorem 7.

Symbol	Meaning	Remark
S	TM space bound	—
k	Batch size	—
t	Transition length	$\leq T$
$(x_{i,\mathsf{start}}, x_{i,\mathsf{end}})$	i-th transition claim	$i \in [k]$
$\boldsymbol{x} = ((x_{i,\mathsf{start}}, x_{i,\mathsf{end}}))_{i\in[k]}$	Input batch	$\boldsymbol{x} \in \{0,1\}^{2Sk}$
λ	Splitting parameter	$\Lambda(T,k)$
$x_{i,j}$	TM state after $(\frac{t}{\lambda}) \cdot j$ steps from $x_{i,\mathsf{start}}$	$j \in [0,\lambda]$
$\boldsymbol{M} = (x_{i,j})_{i\in[k],j\in[0,\lambda]}$	Midpoint matrix	$x_{i,0} = x_{i,\mathsf{start}}, x_{i,\lambda} = x_{i,\mathsf{end}}$
σ	Soundness parameter	—
$\mathbb{F}$	A binary field	specified in Protocol 1
$\chi = \mathsf{cksum}_d(\boldsymbol{M})$	Checksum of the midpoint matrix	$\chi \in \mathbb{F}^{R_{\mathsf{cksum}}\times S(\lambda+1)}$
$t^{\nwarrow} = t/\lambda$	Length of sub-transitions	—
$k^{\nwarrow} = k\lambda$	Expanded batch size	—
$\boldsymbol{x}^{\nwarrow} = (x_{i,j})_{(i,j)\in[k]\times[0,\lambda-1]}$	Expanded batch	$\boldsymbol{x}^{\nwarrow} \in \{0,1\}^{k^{\nwarrow}\times 2S}$
$k^{\downarrow} = \lvert\mathcal{Q}\rvert$	Reduced batch size	$\approx k/d$
$\boldsymbol{x}^{\downarrow} = \boldsymbol{x}[\mathcal{Q},:]$	Reduced batch	$\boldsymbol{x}^{\downarrow} \in \{0,1\}^{k^{\downarrow}\times 2S}$

Table 1: **Summary of Symbols.**

We present the two protocols, SmallBatch and Batch, in Protocol 1 and 2, respectively. In SmallBatch, the prover computes the midpoint matrix, $\boldsymbol{M} \in \{0,1\}^{k\times S(\lambda+1)}$, but given that k is small, the prover can afford to send it in the clear to the verifier. They then make the recursive call to $\mathsf{Batch}(t/\lambda, k\cdot\lambda)$ on the expanded batch $\boldsymbol{x}^{\nwarrow}$. At the end, the verifier performs the resulting checks explicitly on $\boldsymbol{M}$. The general $\mathsf{Batch}(t,k)$ protocol in Protocol 1 uses the auxiliary circuits defined in propositions 2 to 5.

Proposition 2 (The input predicate Φ_{base} for $\mathcal{L}_t^k$)**.** *Consider parameters $n, t = t(n), k = k(n), S = S(n) \in \mathbb{N}$, and a batch of statements $\boldsymbol{x} \in \{0,1\}^{k\times(2S)}$. The predicate Φ_{base} with description $\langle\Phi_{\mathsf{base}}\rangle = (t,k)$ verifies that $\boldsymbol{x} \in \mathcal{L}_t^k$. The implementation circuit C_{base} for Φ_{base} satisfies*

- $\mathsf{size}(C_{\mathsf{base}}) = O(t\cdot k\cdot S)$.
- $\mathsf{depth}(C_{\mathsf{base}}) = O(t + \log k + \log S)$.

Proposition 3 (The output predicate Ψ_{Batch} of Batch)**.** *Consider parameters $n, k, S = S(n), d = d(n), \lambda = \lambda(n) \in \mathbb{N}$, $R_{\mathsf{cksum}} = 2d$, and a boundary checksum slice $\chi_{\mathsf{bdry}} \in \mathbb{F}^{R_{\mathsf{cksum}}\times 2S}$. Let $\mathcal{Q} \subset [k]$ be a subset of indices, and let its implementation circuit be G. Given an instance $\boldsymbol{x} \in \{0,1\}^{k\times(2S)}$ of $\mathcal{L}_t$, a predicate description $\langle\Psi_{\downarrow}\rangle$ for $\Psi_{\downarrow} : \{0,1\}^{|\mathcal{Q}|\times(2S)} \to \{0,1\}$ whose implementation circuit is $C_{\downarrow}$, the predicate Ψ_{Batch} with description $\langle\Psi_{\mathsf{Batch}}\rangle = (\chi_{\mathsf{bdry}}, \langle\mathcal{Q}\rangle, \langle\Psi_{\downarrow}\rangle)$ (where $\Psi_{\downarrow} : \{0,1\}^{|\mathcal{Q}|\times(2S)} \to \{0,1\}$) checks the following.*

1. $\mathsf{cksum}_d(\boldsymbol{x}) = \chi_{\mathsf{bdry}}$.
2. *Expand $\langle\mathcal{Q}\rangle$ using G and compute $\boldsymbol{x}^{\downarrow} := \boldsymbol{x}[\mathcal{Q},:]$.*
3. *Verify that $\Psi_{\downarrow}(\boldsymbol{x}^{\downarrow}) = 1$.*

Protocol 1 Protocol $\mathsf{Batch}(t,k) = (\mathsf{P}(\boldsymbol{x}), \mathsf{V})_{[S,T,t,k,\langle\Phi\rangle]}$ for $\mathcal{L}_t^k[\Phi]$.

Input Parameters: $t, k, T \in \mathbb{N}$, $\langle\Phi\rangle$ is the description of a predicate $\Phi : \{0,1\}^{k\times S(\lambda+1)} \to \{0,1\}$ with implementation circuit C.
Input Batch: $\boldsymbol{x} \in \{0,1\}^{2S\cdot k}$.
Derived Parameters: $\lambda = \Lambda(T,k)$, $N(T,k)$ and σ_{loc} as above, $t_{\mathsf{base}} = \lambda$, $d = 96\sigma_{\mathsf{loc}}\lambda\log(Tk)$, $k_{\mathsf{base}} = d^2$, $\mathbb{F}$ is the smallest binary field such that $|\mathbb{F}| \geq c_0 2^{4\sigma_{\mathsf{loc}}} T^{10}(k+1)^4 S^{c_1} \cdot (\mathsf{depth}(C)\log\mathsf{size}(C)+1)$ for constants c_0, c_1 analyzed in Section 5.2.1. These parameters are *global parameters*: they only depend on (T,k) in the root call to $\mathsf{Batch}(T,k)$.
Verifier Output: $\langle\Psi\rangle \in \{0,1\}^*$, describing a predicate $\Psi : \{0,1\}^{2S\cdot k} \to \{0,1\}$.

(0) **Base Case Handling**

if $t < t_{\mathsf{base}}$ **then**
  Let Φ_{base} be the low-depth circuit that checks $\boldsymbol{x} \in \mathcal{L}_t^k$ (Proposition 2).
  Apply $\mathsf{GKR}_{[\mathbb{F},\langle\Phi_{\mathsf{base}}\rangle]}$ (Theorem 3) and **return** its output.
else if $k < k_{\mathsf{base}}$ **then**
  Call $\mathsf{SmallBatch}_{[S,T,t,k,\langle\Phi\rangle]}$ (Lemma 2) and **return** its output.

(1) **Midpoint Expansion & Checksum**

P finds the matrix $\boldsymbol{M} \in \{0,1\}^{k\times S(\lambda+1)}$ containing all the midpoints from the tableau of $\mathcal{M}$.
P sends the checksum $\chi = \mathsf{cksum}_d(\boldsymbol{M})$ to V.

(2) **Recursive Call on the Expanded Batch**

Let $\boldsymbol{x}^{\nwarrow}$ be the expanded batch of claims, of dimension $(t^{\nwarrow}, k^{\nwarrow}) \coloneqq (\frac{t}{\lambda}, k\lambda)$ in $\boldsymbol{M}$.
Call $\mathsf{Batch}(t^{\nwarrow}, k^{\nwarrow}) = (\mathsf{P}(\boldsymbol{x}^{\nwarrow}), \mathsf{V})_{[\mathbb{F},n,t^{\nwarrow},k^{\nwarrow},\top]}$ on the expanded instance $\boldsymbol{x}^{\nwarrow}$ to ensure that all intermediate states are locally consistent, to obtain $\langle\Psi_{\nwarrow}\rangle$.

(3) **Interactive Proof of Proximity with Row Reduction**

Let Φ_{reduce} be the circuit that verifies $\mathsf{cksum}_d(\boldsymbol{M}) = \chi$, $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}(\boldsymbol{M})) = 1$ and $\Phi(\boldsymbol{M}) = 1$.
Apply $\mathsf{IPP}_{[\sigma_{\mathsf{loc}},\mathbb{F},d,\langle\Phi_{\mathsf{reduce}}\rangle]}$ (Theorem 4) to reduce checking $\Phi_{\mathsf{reduce}}(\boldsymbol{M})$ to $\Psi_{\mathsf{IPP}}(\boldsymbol{M}[\mathcal{Q},:])$.

(4) **Recursive Call on the Reduced Batch & Post-Processing**

Let $\boldsymbol{x}^{\downarrow} \coloneqq \boldsymbol{x}[\mathcal{Q},:]$.
Call $\mathsf{Batch}(t, k^{\downarrow}) = (\mathsf{P}(\boldsymbol{x}^{\downarrow}), \mathsf{V})_{[\mathbb{F},n,t,|\mathcal{Q}|,\langle\Psi_{\mathsf{IPP}}\rangle]}$ on $\boldsymbol{x}^{\downarrow} = \boldsymbol{x}[\mathcal{Q},:]$ to obtain $\langle\Psi_{\downarrow}\rangle$
return $\langle\Psi_{\mathsf{Batch}}\rangle$, the description of the circuit in Proposition 3 that checks $\mathsf{cksum}_d(\boldsymbol{x}[:,0:S]) = \chi[:,0:S], \mathsf{cksum}_d(\boldsymbol{x}[:,-S:]) = \chi[:,-S:]$ and that $\Psi_{\downarrow}(\boldsymbol{x}[\mathcal{Q},:]) = 1$.

Protocol 2 Protocol $\mathsf{SmallBatch} = (\mathsf{P}_{\mathsf{SmallBatch}}(\boldsymbol{x}), \mathsf{V}_{\mathsf{SmallBatch}})_{[S,T,t,k,\langle\Phi\rangle]}$ for $k < k_{\mathsf{base}}$.

Input Parameters: $t, k, T \in \mathbb{N}$, $\langle\Phi\rangle$ is the description of a predicate $\Phi : \{0,1\}^{k\times S(\lambda+1)} \to \{0,1\}$ with implementation circuit C. **Input Batch:** $\boldsymbol{x} \in \{0,1\}^{2S\cdot k}$.
Derived Parameters: $\lambda = \Lambda(T,k)$, $N(T,k)$ and σ_{loc} as above, $t_{\mathsf{base}} = \lambda$, $d = 96\sigma_{\mathsf{loc}}\lambda\log(Tk)$, $k_{\mathsf{base}} = d^2$, $\mathbb{F}$ is the smallest binary field such that $|\mathbb{F}| \geq c_0 2^{4\sigma_{\mathsf{loc}}} T^{10}(k+1)^4 S^{c_1}\cdot(\mathsf{depth}(C)\log\mathsf{size}(C)+1)$ for constants c_0, c_1 analyzed in Section 5.2.1.
Verifier Output: $\langle\Psi\rangle \in \{0,1\}^*$, describing a predicate $\Psi : \{0,1\}^{2S\cdot k} \to \{0,1\}$.
Assumption: $k < k_{\mathsf{base}}$ is small.

(1) **Midpoint Expansion**

P finds the matrix $\boldsymbol{M} \in \{0,1\}^{k\times S(\lambda+1)}$ containing all the midpoints and sends $\boldsymbol{M}$ to V.

(2) **Recursive Call on the Expanded Batch**

Call $\mathsf{Batch}(t^{\nwarrow}, k^{\nwarrow}) = (\mathsf{P}(\boldsymbol{x}^{\nwarrow}), \mathsf{V})_{[\mathbb{F},n,t^{\nwarrow},k^{\nwarrow},\top]}$ on the expanded instance $\boldsymbol{x}^{\nwarrow}$ to obtain $\langle\Psi_{\nwarrow}\rangle$.

(3) **Explicit Checks and Post-processing**

V checks $C_{\nwarrow}(\boldsymbol{x}^{\nwarrow}, \langle\Psi_{\nwarrow}\rangle) = 1$ and $\Phi(\boldsymbol{M}) = 1$. It **rejects** otherwise.
V constructs $\boldsymbol{x}^{\nwarrow}$ from $\boldsymbol{M}$.
return $\langle\Psi_{\mathsf{SmallBatch}}\rangle$, the description of the circuit in Proposition 5 that verifies the boundaries of $\boldsymbol{x}^{\nwarrow}$ agree with $\boldsymbol{x}$ and that $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$.

Note that $|\langle\Psi_{\mathsf{Batch}}\rangle| = |\langle\Psi_{\downarrow}\rangle| + |\langle\mathcal{Q}\rangle| + O(R_{\mathsf{cksum}} S \log|\mathbb{F}|) = |\langle\Psi_{\downarrow}\rangle| + |\langle\mathcal{Q}\rangle| + O(dS\log|\mathbb{F}|)$. *With* $\widetilde{O}$ *hiding* $\mathrm{polylog}(n, Sk, d, \lambda)$ *factors, the implementation circuit* C_{Batch} *for* Ψ_{Batch} *satisfies*

- $\mathsf{size}(C_{\mathsf{Batch}}) = \widetilde{O}(kR_{\mathsf{cksum}}S + |\mathcal{Q}|(\mathsf{size}(G) + S) + \mathsf{size}(C_{\downarrow}))$.
- $\mathsf{depth}(C_{\mathsf{Batch}}) = \widetilde{O}(1) + \mathsf{depth}(G) + \mathsf{depth}(C_{\downarrow})$.

Proposition 4 (The input predicate Φ_{reduce} for the IPP)**.** *Consider parameters* $n, S = S(n), k = k(n), d = d(n), \lambda = \lambda(n) \in \mathbb{N}$. *Given a checksum* $\chi \in \mathbb{F}^{R_{\mathsf{cksum}} \times (S(\lambda+1))}$, *a predicate description* $\langle\Phi\rangle$ *for* $\Phi : \mathbb{F}^{k\times(S(\lambda+1))} \to \{0,1\}$ *with implementation circuit* C, *a matrix* $\boldsymbol{M} \in \{0,1\}^{k\times(S(\lambda+1))}$, *and a predicate description* $\langle\Psi_{\nwarrow}\rangle$ *for* $\Psi_{\nwarrow} : \{0,1\}^{(k\lambda)\times(2S)} \to \{0,1\}$ *with implementation circuit* $C_{\nwarrow}$, *the predicate* Φ_{reduce} *with description* $\langle\Phi_{\mathsf{reduce}}\rangle = (\chi, \langle\Phi\rangle, \langle\Psi_{\nwarrow}\rangle, d, \lambda)$ *performs the following on input* $\boldsymbol{M}$*:*

1. *Reads out* $\boldsymbol{x}^{\nwarrow} = ((\boldsymbol{x}^{\nwarrow}{}_{i,j}, \boldsymbol{x}^{\nwarrow}{}_{i,j+1}))_{i,j\in[k]\times[0,\lambda-1]} \in \{0,1\}^{(k\cdot\lambda)\times(2S)}$ *from* $\boldsymbol{M}$ *and verifies that* $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$.
2. *Verifies that* $\mathsf{cksum}_d(\boldsymbol{M}) = \chi$.
3. *Verifies that* $\Phi(\boldsymbol{M}) = 1$.

With C *denoting* Φ*'s implementation circuit and* $\widetilde{O}$ *hiding* $\mathrm{polylog}(n, S, k, d, \lambda)$ *factors, the implementation circuit* C_{reduce} *for* Φ_{reduce} *satisfies*

- $\mathsf{size}(C_{\mathsf{reduce}}) = \widetilde{O}(kR_{\mathsf{cksum}}S\lambda + \mathsf{size}(C) + \mathsf{size}(C_{\nwarrow}))$.
- $\mathsf{depth}(C_{\mathsf{reduce}}) = \widetilde{O}(\max(\mathsf{depth}(C), \mathsf{depth}(C_{\nwarrow})))$.

Proposition 5 (The output predicate $\Psi_{\mathsf{SmallBatch}}$ of $\mathsf{SmallBatch}$)**.** *Consider parameters* $n, S = S(n), k = k(n), \lambda = \lambda(n) \in \mathbb{N}$. *Let* $\boldsymbol{x} \in \{0,1\}^{k\times(2S)}$ *be a batch of statements and* $\boldsymbol{x}^{\nwarrow} = ((x_{i,j}, x_{i,j+1}))_{(i,j)\in[k]\times[0,\lambda-1]} \in \{0,1\}^{(k\cdot\lambda)\times(2S)}$ *be the batch of midpoint statements for* $\boldsymbol{x}$ *(given by the midpoint matrix* $\boldsymbol{M}_{\boldsymbol{x}}$ *for parameter* λ*). Given a description* $\Psi_{\nwarrow}$ *for a predicate* $\Psi_{\nwarrow} : \{0,1\}^{(k\lambda)\times(2S)} \to \{0,1\}$ *whose implementation circuit is* $C_{\nwarrow}$, *we let* $\langle\Psi_{\mathsf{SmallBatch}}\rangle = (\boldsymbol{x}^{\nwarrow}, \langle\Psi_{\nwarrow}\rangle)$. *The predicate* $\Psi_{\mathsf{SmallBatch}}(\boldsymbol{x})$ *verifies the following:*

1. *The boundary states of* $\boldsymbol{x}^{\nwarrow}$ *agree with* $\boldsymbol{x}$ *(i.e.,* $x_{i,0} = x_{i,\mathsf{start}}$ *and* $x_{i,\lambda} = x_{i,\mathsf{end}}$ *for all* $i \in [k]$*).*
2. *The condition* $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$ *holds.*

Note that $|\langle\Psi_{\mathsf{SmallBatch}}\rangle| = O(k\lambda S + |\langle\Psi_{\nwarrow}\rangle|)$. *With* $\widetilde{O}$ *hiding* $\mathrm{poly}(n, S, k, \lambda)$ *factors, the implementation circuit* $C_{\mathsf{SmallBatch}}$ *for* $\Psi_{\mathsf{SmallBatch}}$ *satisfies*

- $\mathsf{size}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(kS\lambda) + \mathsf{size}(C_{\nwarrow})$.
- $\mathsf{depth}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(1) + \mathsf{depth}(C_{\nwarrow})$.

5.2 Analysis of Our Construction

We show how to prove Theorem 7 and lemma 2 simultaneously, assuming Propositions 2 to 5, whose proofs are deferred to Section 5.3.

By strong induction, suppose for all (t', k') such that either $t' < t$ or $t' = t$ and $k' < k$, there exists a protocol $\mathsf{Batch}(t', k')$ with the stated properties.

5.2.1 Completeness and parameter selection

Completeness follows from the completeness of the underlying sub-protocols.

We set $\lambda = \left\lceil 2^{\sqrt{\log\binom{2\log T+\log k}{\log T}}} \right\rceil$, $\sigma_{\mathsf{loc}} = \sigma + \lceil \log(4N(T,k)) \rceil + 2$, $d = 96\sigma_{\mathsf{loc}}\lambda \log(Tk)$, $t_{\mathsf{base}} = \lambda$ and $k_{\mathsf{base}} = d^2$. The discussion on why this choice minimizes the overall complexity of the protocol appears in Section 5.2.3.

Here we discuss the selection of the field size.

- **In the base case ($t < t_{\mathsf{base}} = \lambda$):**

  We invoke the GKR protocol on C_{base} with $\mathsf{size}(C_{\mathsf{base}}) = O(t \cdot k \cdot S)$, $\mathsf{depth}(C_{\mathsf{base}}) = O(\lambda + \log k + \log S)$, and thus $\mathsf{depth}(C_{\mathsf{base}}) \cdot \log \mathsf{size}(C_{\mathsf{base}}) = \lambda(\log(\lambda k S))^2$. Therefore, to make the soundness error less than $2^{-\sigma_{\mathsf{loc}}}$, we set

  $$|\mathbb{F}| > 2^{\sigma_{\mathsf{loc}}} \cdot C_{\mathsf{GKR}}\lambda \cdot (\log(\lambda(k+1)S))^2 \in O(2^{\sigma_{\mathsf{loc}}} T^3 k^2 S).$$

- **In the general case ($t \geq t_{\mathsf{base}}$, $k \geq k_{\mathsf{base}}$):**

  For some absolute constant c_2, the following bound follows from Claim 4 in Section 5.2.3,

  $$\begin{aligned}\mathsf{depth}(C_{\mathsf{reduce}}) \log \mathsf{size}(C_{\mathsf{reduce}}) &\leq \widetilde{O}(\mathsf{depth}(C_{t,k}) \log \mathsf{size}(C_{t,k})) \\ &= \max(\mathsf{depth}(C) \log \mathsf{size}(C), \lambda) \log(|\mathbb{F}|nST)^{c_2},\end{aligned}$$

  where $C_{t,k}$ is the implementation circuit of $\Phi_{t,k}$, the input predicate defining $\mathcal{L}_t^k[\Phi_{t,k}]$ in the recursive call to $\mathsf{Batch}(t,k)$, and C_{reduce} is the implementation circuit for the predicate on which we run IPP.

  Let $\Gamma \coloneqq \mathsf{depth}(C) \log \mathsf{size}(C) + 1$. In order to apply the soundness guarantee in Theorem 4, the field size must be at least (recalling that C_{GKR} and c are some absolute constants),

  $$\begin{aligned}&C_{\mathsf{GKR}} \cdot 2^{\sigma_{\mathsf{loc}}+4c+7} \cdot ((\sigma_{\mathsf{loc}} dS(\lambda+1)\log(|\mathbb{F}|(k+1)))^c + \mathsf{depth}(C_{\mathsf{reduce}}) \log \mathsf{size}(C_{\mathsf{reduce}}) + 1) \\ \in &O(2^{2\sigma_{\mathsf{loc}}}(\lambda^{c+1} S^c \log k^c + \max(\Gamma, \lambda) \log(|\mathbb{F}|nSTk)^{c_2})) \\ \subset &O(2^{4\sigma_{\mathsf{loc}}} T^5 k^4 \Gamma \cdot (S \log |\mathbb{F}|)^{\max(c,c_2)+1}).\end{aligned}$$

  Note that we used the actual IPP invocation parameter σ_{loc}, $d = 96\sigma_{\mathsf{loc}}\lambda\log(Tk)$, and $(\lambda \log(nT(k+1)))^{O(1)} = o(Tk)$ to simplify the bounds.

To summarize, there exist some constants c_0, c_2 such that picking

$$|\mathbb{F}| \geq c_0 2^{4\sigma_{\mathsf{loc}}} T^{10}(k+1)^4 S^{c_1} \cdot (\mathsf{depth}(C) \log \mathsf{size}(C) + 1)$$

would satisfy both requirements.[5]

[5] Specifically, c_0 is the hidden constant in the O notation in the general case above and $c_1 = 2(\max(c, c_2) + 1)$. Note that we also used the fact that $|\mathbb{F}|^{1/2} \gg \text{polylog}|\mathbb{F}|$ when $|\mathbb{F}|$ is large.

5.2.2 Soundness.

Suppose $\boldsymbol{x} \notin \mathcal{L}_t^k[\Phi]$ and let P^* be a cheating prover. Let $\langle \Psi \rangle := \langle \mathsf{P}, \mathsf{V}^*(\boldsymbol{x}) \rangle_{[\mathbb{F},\langle\Phi\rangle,n,t,k,T]}$ be the output of either $\mathsf{Batch}(t,k)$ if $k > k_{\mathsf{base}}$ or $\mathsf{SmallBatch}(t,k)$ if $k \leq k_{\mathsf{base}}$, and E be the event that $\Psi(\boldsymbol{x}) = 1$. Let $\epsilon(t,k)$ be the soundness error of $\mathsf{Batch}(t,k)$. Our goal is to give an upper bound for $\Pr[E] \leq \epsilon(t,k)$.

Claim 1. *If $t < t_{\mathsf{base}}$, $\epsilon < 2^{-\sigma_{\mathsf{loc}}}$.*

Proof. In this case, the verifier simply returns the output of the GKR protocol for checking $\Phi_{\mathsf{base}}(\boldsymbol{x})$, which verifies that $\boldsymbol{x} \in \mathcal{L}_t^k$. By the soundness of GKR (Theorem 3), $\epsilon < 2^{-\sigma_{\mathsf{loc}}}$. □

Therefore, we consider the case when $t \geq t_{\mathsf{base}}$. Let $\boldsymbol{M}$ denote the midpoint matrix of $\boldsymbol{x}$. We have the following cases depending on whether the batch size k is small:

- In protocol $\mathsf{SmallBatch}$, some matrix $\boldsymbol{M}^*$ is sent explicitly to the verifier.
- In protocol Batch, let χ^* be the checksum sent by P^*. Since the checksum function is unique-decoding up to d deviations, χ^* uniquely defines at most one matrix $\boldsymbol{M}^* \in \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M})$.

In either case, we can define the matrix $\boldsymbol{M}^* \in \{0,1\}^{k \times S(\lambda+1)}$. If such a matrix exists, define it uniquely; otherwise, set $\boldsymbol{M}^* = \boldsymbol{M}$ arbitrarily. Let $\boldsymbol{x}^{\nwarrow *}$ be the instance of $\mathcal{L}_{t^{\nwarrow}}^{k^{\nwarrow}}$ read from $\boldsymbol{M}^*$. In what follows, if an event A is characterized by a logical statement, we may use A for both the event and the logical statement.

Fact 1. *Suppose $\boldsymbol{x} \notin \mathcal{L}_t^k[\Phi]$, and define F_1, F_2, and F_3 as follows.*

(F_1) $\Phi(\boldsymbol{M}^*) = 1$

(F_2) $\boldsymbol{M}^*[:, : S] = \boldsymbol{x}[:, : S]$ *and* $\boldsymbol{M}^*[:, -S :] = \boldsymbol{x}[:, -S :]$*, i.e. the leftmost start states and rightmost end states of $\boldsymbol{M}^*$ are the same as those in $\boldsymbol{x}$*

(F_3) $\boldsymbol{x}^{\nwarrow *} \in \mathcal{L}_{t^{\nwarrow}}^{k^{\nwarrow}}$.

Then $\Pr[F_1 \cap F_2 \cap F_3] = 0$*, i.e.* $\Pr[\neg F_1 \cup \neg F_2 \cup \neg F_3] = 1$.

Proof. It suffices to show that $F_2 \wedge F_3 \implies \neg F_1$. We have that $F_2 \wedge F_3$ implies $\boldsymbol{M} = \boldsymbol{M}^*$ and $\boldsymbol{x}^{\nwarrow} = \boldsymbol{x}^{\nwarrow *}$ because $\mathcal{M}$ is deterministic: (a) fixing the start states and (b) ensuring the transitions are correct would uniquely determine the entire path. On the other hand, $\boldsymbol{x} \notin \mathcal{L}_t^k[\Phi] \Leftrightarrow \boldsymbol{x}^{\nwarrow} \notin \mathcal{L}_{t^{\nwarrow}}^{k^{\nwarrow}} \vee \Phi(\boldsymbol{M}) = 0$. Since $\boldsymbol{x}^{\nwarrow} \in \mathcal{L}_{t^{\nwarrow}}^{k^{\nwarrow}}$ (by F_3 and the fact that $\boldsymbol{x}^{\nwarrow} = \boldsymbol{x}^{\nwarrow *}$), we must have $\Phi(\boldsymbol{M}) = 0$. □

Therefore, by a union bound:

$$\Pr[E] \leq \Pr[E \cap \neg F_1] + \Pr[E \cap \neg F_2] + \Pr[E \cap \neg F_3].$$

Let G be the event that $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow *}) = 1$. By the inductive hypothesis, the protocol $\mathsf{Batch}(t^{\nwarrow}, k^{\nwarrow})$ at Item (2) in both versions of the protocol is sound, so

Fact 2. *In both versions of the protocol, Batch and SmallBatch,* $\Pr[G \mid \neg F_3] = \Pr[\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow *}) = 1 \mid \boldsymbol{x}^{\nwarrow *} \notin \mathcal{L}_{t^{\nwarrow}}^{k^{\nwarrow}}] \leq \epsilon(t^{\nwarrow}, k^{\nwarrow})$.

Moreover, by a union bound,

$$\begin{aligned}\Pr[E] &\le \Pr[E \cap (\neg F_1 \cup \neg F_2 \cup \neg F_3)] \\ &\le \Pr[E \cap \neg F_1] + \Pr[E \cap \neg F_2] + \Pr[E \cap \neg F_3]. \end{aligned} \tag{1}$$

Now we can analyze the two versions of the protocol.

Claim 2. *In SmallBatch,* $\Pr[E] \le \epsilon(t^{\nwarrow}, k^{\nwarrow})$.

Proof. The verifier has access to the entire matrix $\boldsymbol{M}^*$, and the output Ψ is the circuit $\Psi_{\nwarrow}$ in Proposition 5, which checks $F_2 \wedge G$.

- Since the verifier checks that $\Phi(\boldsymbol{M}^*) = 1$ (*i.e.* F_1) in Item (3) in SmallBatch, if $\neg F_1$ holds, it rejects on that line, and thus $\Pr[E \mid \neg F_1] = 0$.
- Since the output circuit Ψ produced by SmallBatch checks $\boldsymbol{M}^*[:, 0 : S] = \boldsymbol{x}[:, 0 : S]$ (*i.e.* F_2), and also $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$ (*i.e.* G), $\Pr[E \mid \neg F_2] = \Pr[E \mid \neg G] = 0$.

Therefore, by Equation (1) and fact 2,

$$\begin{aligned}\Pr[E] &\le 0 + \Pr[E \wedge \neg F_2] + \Pr[E \wedge \neg F_3] \\ &\le 0 + \Pr[E \mid \neg F_2] + \Pr[E \wedge \neg F_3 \wedge \neg G] + \Pr[E \wedge \neg F_3 \wedge G] \\ &\le 2 \cdot 0 + \Pr[E \mid \neg G] + Pr[G \mid \neg F_3] \le \epsilon(t^{\nwarrow}, k^{\nwarrow}). \end{aligned}$$

□

Claim 3. *In Batch,* $\Pr[E] \le \epsilon(t, k^{\downarrow}) + \epsilon(t^{\nwarrow}, k^{\nwarrow}) + 2^{-\sigma_{\mathsf{loc}}}$.

Proof. We first prove a series of facts.

Fact 3. *In Batch,* $\Pr[E \mid \neg F_2] = 0$.

Proof of Fact 3. Recall that $\neg F_2$ is the event that $\boldsymbol{M}^*[:, 0 : S] \neq \boldsymbol{x}[:, 0 : S]$ or $\boldsymbol{M}^*[:, -S :] \neq \boldsymbol{x}[:, -S :]$. If $\boldsymbol{M}^*[:, 0 : S] \neq \boldsymbol{x}[:, 0 : S]$, by unique decoding (Lemma 1), $\mathsf{cksum}(\boldsymbol{x}[:, 0 : S]) \neq \chi^*[:, 0 : S] = \mathsf{cksum}(\boldsymbol{M}^*[:, 0 : S])$. The output circuit Ψ produced by Batch checks that $\mathsf{cksum}(\boldsymbol{x}[:, 0 : S]) = \chi^*[:, 0 : S]$ (Proposition 3), so indeed $\Pr[E \mid \neg F_2] = 0$. The same argument applies for $\boldsymbol{M}^*[:, -S :] \neq \boldsymbol{x}[:, -S :]$. □

Let $k^{\downarrow} := |\mathcal{Q}|$, $\boldsymbol{M}^{\downarrow} := \boldsymbol{M}[\mathcal{Q}, :]$ be the corresponding variables after the protocol runs Item (3).

Fact 4. $\Pr[\Psi_{IPP}(\boldsymbol{M}^{\downarrow}) = 1 \mid \neg(F_1 \cap G)] \le 2^{-\sigma_{\mathsf{loc}}}$.

Proof of Fact 4. By the unique decoding property (Lemma 1), $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) \cap {\Phi_{\mathsf{reduce}}}^{-1}(1) \subset \{\boldsymbol{M}^*\}$ because Φ_{reduce} checks that its input has checksum χ^*. However, Φ_{reduce} also checks F_1 and G, so $\boldsymbol{M}^*$ is excluded from its acceptance range when $\neg(F_1 \wedge G)$, and $\boldsymbol{M}$ is d-Δ_c-far from $\mathcal{L}_{\Phi_{\mathsf{reduce}}}$.

The prerequisite of the IPP soundness is also satisfied by our choice of d and $\mathbb{F}$, so Theorem 4 entails that $\Pr[\Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 1 \mid \neg(F_1 \cap G)] \le 2^{-\sigma_{\mathsf{loc}}}$. □

Let $\langle \Psi_{\downarrow} \rangle$ be the output of Item (4) in Batch, and $\boldsymbol{x}^{\downarrow} := \boldsymbol{x}[\mathcal{Q}, :]$. Applying the inductive hypothesis to the protocol $\mathsf{Batch}(t, k^{\downarrow})$, we have

$$\Pr[E \mid \Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 0] = \Pr[\Psi_{\downarrow}(\boldsymbol{x}^{\downarrow}) = 1 \mid \Psi_{\downarrow}(\boldsymbol{M}^{\downarrow}) = 0] \le \epsilon(t, k^{\downarrow}).$$

This, along with Fact 4, gives

$$\begin{aligned}\Pr[E \mid \neg(F_1 \cap G)] &\le \Pr[E \mid \Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 0] + \Pr[\Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 1 \mid \neg(F_1 \cap G)] \\ &\le \epsilon(t, k^{\downarrow}) + 2^{-\sigma_{\mathsf{loc}}}.\end{aligned}$$

Finally, combining with Facts 2 and 3 and equation (1),

$$\begin{aligned}\Pr[E] &\le \Pr[E \cap (\neg F_1 \cup \neg F_3)] + \Pr[E \cap \neg F_2] \\ &\le \Pr[E \cap (\neg F_3 \cap G)] + \Pr[E \cap (\neg F_1 \cup (\neg F_3 \cap \neg G))] + \Pr[E \mid \neg F_2] \\ &\le \Pr[G \mid \neg F_3] + \Pr[E \mid \neg F_1 \cup \neg G] + 0 \\ &\le \epsilon(t, k^{\downarrow}) + \epsilon(t^{\nwarrow}, k^{\nwarrow}) + 2^{-\sigma_{\mathsf{loc}}}.\end{aligned}$$

□

To sum up the three cases,

$$\Pr[E] \le \begin{cases} 2^{-\sigma_{\mathsf{loc}}} & \text{if } t < t_{\mathsf{base}}, \\ \epsilon(t^{\nwarrow}, k^{\nwarrow}) & \text{if } k < k_{\mathsf{base}}, \\ \epsilon(t, k^{\downarrow}) + \epsilon(t^{\nwarrow}, k^{\nwarrow}) + 2^{-\sigma_{\mathsf{loc}}}. & \text{otherwise.} \end{cases}$$

It remains to plug in our choice of parameters. Let $\mathcal{T}(t,k)$ be the recursion tree generated by the two recursive calls. By Section 5.2.3, $|\mathcal{T}(t,k)| \le N(T,k)$. Every base node contributes at most $2^{-\sigma_{\mathsf{loc}}}$, and every general node contributes at most $2^{-\sigma_{\mathsf{loc}}}$ in addition to its recursive errors. Thus the inductive recurrence above gives

$$\epsilon(t,k) \le 2|\mathcal{T}(t,k)| \cdot 2^{-\sigma_{\mathsf{loc}}} \le 2N(T,k) \cdot 2^{-\sigma - \lceil \log(4N(T,k)) \rceil - 2} < 2^{-\sigma}.$$

5.2.3 Complexities.

Let $\langle \Psi \rangle$ be output by the protocol, and C_Ψ be the predicate's implementation circuit. Let $\ell(t,k)$, $a(t,k)$, $\mathsf{Ptime}(t,k)$, $\mathsf{Vtime}(t,k)$ be the round complexity, per-round communication complexity, prover runtime, and verifier time of $\mathsf{Batch}(t,k)$, and $\mathsf{size}(t,k)$, $\mathsf{depth}(t,k)$, $|\langle \Psi \rangle|(t,k)$ be the size, depth and description length of the output predicate of $\mathsf{Batch}(t,k)$.

$\widetilde{O}$ hides $\mathrm{polylog}(|\mathbb{F}|, n, S, T, k)$ factors. Recall that $t^{\nwarrow} = t/\lambda$, $k^{\nwarrow} = k \cdot \lambda$, $k^{\downarrow} = |\mathcal{Q}| \le \lceil 24\sigma_{\mathsf{loc}} \frac{k}{d} \rceil$.

Claim 4. *The size, depth, and description length of the output low-depth predicate satisfy:*

$$\mathsf{size}(t,k) = \begin{cases} \widetilde{O}(k \cdot S) & \textit{if } t < t_{\mathsf{base}}, \\ \mathsf{size}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} S \lambda) & \textit{if } k < k_{\mathsf{base}}, \\ \mathsf{size}(t, k^{\downarrow}) + \widetilde{O}(k \cdot \mathrm{poly}(d) S) & \textit{otherwise,} \end{cases}$$

$$\mathsf{depth}(t,k) = \begin{cases} \widetilde{O}(t_{\mathsf{base}}) & \textit{if } t < t_{\mathsf{base}}, \\ \mathsf{depth}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(1) & \textit{if } k < k_{\mathsf{base}}, \\ \mathsf{depth}(t, k^{\downarrow}) + \widetilde{O}(1) & \textit{otherwise,} \end{cases}$$

$$|\langle \Psi \rangle|(t,k) = \begin{cases} \widetilde{O}(1) & \textit{if } t < t_{\mathsf{base}}, \\ |\langle \Psi \rangle|(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} \cdot S\lambda) & \textit{if } k < k_{\mathsf{base}}, \\ |\langle \Psi \rangle|(t, k^{\downarrow}) + \widetilde{O}(dS + \mathrm{poly}(d)) & \textit{otherwise,} \end{cases}$$

Note that $t_{\mathsf{base}} = \lambda$ *and* $k_{\mathsf{base}} = d^2$. *Along any recursion path, at most* $\log_\lambda T$ *expanded-batch calls occur, so every encountered state satisfies* $k' \le kT$. *Consequently* $\mathsf{depth}(t', k') < \widetilde{O}(\lambda)$, $\log \mathsf{size}(t', k') \le \mathrm{polylog}(k, T, S, \lambda, d)$, *and* $|\langle\Psi\rangle|(t', k') < \widetilde{O}(d^2 S\lambda + \mathrm{poly}(d))$ *for all* (t', k') *encountered in the recursion.*

We analyze the complexities (and thereby prove Claim 4) by the following cases.

- When $t < t_{\mathsf{base}}$, by Theorem 3 and proposition 2, we have
  - $\mathsf{size}(C_{\mathsf{base}}) = O(t_{\mathsf{base}} \cdot k \cdot S)$,
  - $\mathsf{depth}(C_{\mathsf{base}}) = O(t_{\mathsf{base}} + \log k + \log S) = \widetilde{O}(t_{\mathsf{base}})$,
  - $\ell(t, k) = \ell_{\mathsf{GKR}}(C_{\mathsf{base}}) = \mathsf{depth}(C_{\mathsf{base}}) \log \mathsf{size}(C_{\mathsf{base}}) = \widetilde{O}(t_{\mathsf{base}})$,
  - $a(t, k) = a_{\mathsf{GKR}}(C_{\mathsf{base}}) = O(\log |\mathbb{F}|) = \widetilde{O}(1)$.
  - $\mathsf{Ptime}(t, k) = \mathsf{Ptime}_{\mathsf{GKR}}(C_{\mathsf{base}}) = \widetilde{O}(\mathrm{poly}(\mathsf{size}(C_{\mathsf{base}}) \cdot \mathrm{polylog}|\mathbb{F}|)) = \mathrm{poly}(t_{\mathsf{base}} k S)$,
  - $\mathsf{Vtime}(t, k) = \mathsf{Vtime}_{\mathsf{GKR}}(C_{\mathsf{base}}) = \widetilde{O}((\ell_{\mathsf{GKR}}(C_{\mathsf{base}}) + |\langle\Phi_{\mathsf{base}}\rangle|)\log |\mathbb{F}|) = \widetilde{O}(t_{\mathsf{base}})$.
  - $\mathsf{size}(t, k) = \widetilde{O}(k \cdot S)$.
  - $\mathsf{depth}(t, k) = \widetilde{O}(1)$.
  - $|\langle\Psi\rangle| = O(\log |\mathbb{F}|) = \widetilde{O}(1)$.
- When $k < k_{\mathsf{base}}$, we run the $\mathsf{SmallBatch}$ protocol, in which case
  - $\ell(t, k) = \ell(t^{\nwarrow}, k^{\nwarrow}) + O(1)$.
  - $a(t, k) = a(t^{\nwarrow}, k^{\nwarrow}) + O(k_{\mathsf{base}} S\lambda)$.
  - $\mathsf{Ptime}(t, k) = \mathsf{Ptime}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} S\lambda)$.
  - $\mathsf{Vtime}(t, k) = \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} S\lambda) + \mathsf{size}(C) \cdot \mathrm{polylog}|\mathbb{F}|$.

  The additional terms in the prover and verifier time are due to the time to send and receive the matrix $\boldsymbol{M}$ and the time to verify the predicate Φ by simulating its implementation circuit C (with a $\mathrm{polylog}|\mathbb{F}|$ overhead). Also note that Ptime does not pay any cost in computing the intermediate states at Item (1) because we assume it has access to the entire tableau as auxiliary input.

  Furthermore, by Proposition 5,
  - $\mathsf{size}(t, k) = \mathsf{size}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(k_{\mathsf{base}} S\lambda) + \mathsf{size}(t^{\nwarrow}, k^{\nwarrow})$.
  - $\mathsf{depth}(t, k) = \mathsf{depth}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(1) + \mathsf{depth}(t^{\nwarrow}, k^{\nwarrow})$.
  - $|\langle\Psi\rangle|(t, k) = \widetilde{O}(k_{\mathsf{base}} S\lambda) + |\langle\Psi_{\nwarrow}\rangle| = \widetilde{O}(k_{\mathsf{base}} S\lambda) + |\langle\Phi\rangle|(t^{\nwarrow}, k^{\nwarrow})$.
- In the general case, let $t^{\nwarrow} = t/\lambda$, $k^{\nwarrow} = k\lambda$, and $k^{\downarrow} = |\mathcal{Q}|$. By Theorem 4, $|\mathcal{Q}| \le \lceil 24\sigma_{\mathsf{loc}} \cdot \frac{k}{d} \rceil < \frac{k}{\lambda}$, where the final inequality follows from $d = 96\sigma_{\mathsf{loc}}\lambda \log(Tk)$. We first work out the bounds for intermediate parameters, using Proposition 4 and Claim 4.
  - $R_{\mathsf{cksum}} = 2d$.
  - $\mathsf{size}(C_{\mathsf{reduce}}) = \widetilde{O}(kR_{\mathsf{cksum}} S\lambda) + \mathsf{size}(C) + \mathsf{size}(C_{\nwarrow})$, and therefore $\log \mathsf{size}(C_{\mathsf{reduce}}) \le \mathrm{polylog}(k, T, S, \lambda, d, |\mathbb{F}|) + \log \mathsf{size}(C)$.

– $\mathsf{depth}(C_{\mathsf{reduce}}) = \widetilde{O}(\max(\mathsf{depth}(C), \mathsf{depth}(C_{\nwarrow}))) = \widetilde{O}(\mathsf{depth}(C))$.

Therefore, $\mathsf{depth}(C_{\mathsf{reduce}}) \log \mathsf{size}(C_{\mathsf{reduce}}) = \widetilde{O}(\mathsf{depth}(C) \log \mathsf{size}(C) + \lambda)$. By Theorem 4,

– $\begin{aligned}\ell(t,k) &= \ell(t^{\nwarrow}, k^{\nwarrow}) + \ell(t, k^{\downarrow}) + \ell_{\mathsf{IPP}}(C_{\mathsf{reduce}}) + O(1)\\ &= \ell(t^{\nwarrow}, k^{\nwarrow}) + \ell(t, k^{\downarrow}) + \widetilde{O}(\mathsf{depth}(C_{\mathsf{reduce}}) \log \mathsf{size}(C_{\mathsf{reduce}}))\\ &= \ell(t^{\nwarrow}, k^{\nwarrow}) + \ell(t, k^{\downarrow}) + \widetilde{O}(\mathsf{depth}(C) \log \mathsf{size}(C) + \lambda).\end{aligned}$

– $\begin{aligned}a(t,k) &= a(t^{\nwarrow}, k^{\nwarrow}) + a(t, k^{\downarrow}) + O(R_{\mathsf{cksum}} S \lambda \log |\mathbb{F}|) + a_{\mathsf{IPP}}(C_{\mathsf{reduce}})\\ &= a(t^{\nwarrow}, k^{\nwarrow}) + a(t, k^{\downarrow}) + \widetilde{O}(S\lambda \mathrm{poly}(d)).\end{aligned}$

– $\begin{aligned}\mathsf{Ptime}(t,k) &= \mathsf{Ptime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Ptime}(t, k^{\downarrow}) + \widetilde{O}(k \cdot R_{\mathsf{cksum}} \cdot S \cdot \lambda \cdot \mathrm{polylog}|\mathbb{F}|)\\ &\quad + \mathsf{Ptime}_{\mathsf{IPP}}(C_{\mathsf{reduce}})\\ &= \mathsf{Ptime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Ptime}(t, k^{\downarrow}) + \widetilde{O}(\mathrm{poly}(kdS\lambda \mathsf{size}(C))).\end{aligned}$

– $\begin{aligned}\mathsf{Vtime}(t,k) &= \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Vtime}(t, k^{\downarrow}) + \mathsf{Vtime}_{\mathsf{IPP}}(C_{\mathsf{reduce}})\\ &= \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Vtime}(t, k^{\downarrow})\\ &\quad + \widetilde{O}((\mathsf{depth}(C_{\mathsf{reduce}}) \log \mathsf{size}(C_{\mathsf{reduce}}) + |\langle \Phi_{\mathsf{reduce}} \rangle|) \cdot Sd\lambda + \mathrm{poly}(d))\\ &= \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Vtime}(t, k^{\downarrow})\\ &\quad + \widetilde{O}((\mathsf{depth}(C) \log \mathsf{size}(C) + \lambda + |\langle \Phi \rangle| + |\langle \Psi_{\nwarrow} \rangle| + dS\lambda) Sd\lambda + \mathrm{poly}(d))\\ &= \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Vtime}(t, k^{\downarrow})\\ &\quad + \widetilde{O}((\mathsf{depth}(C) \log \mathsf{size}(C) + |\langle \Phi \rangle|) S^2 \lambda^2 \mathrm{poly}(d)).\end{aligned}$

  Note that we used Claim 4, *i.e.* that $|\langle \Psi_{\nwarrow} \rangle| \leq d^2 S\lambda + \mathrm{poly}(d)$, and that $k_{\mathsf{base}} = \widetilde{O}(d^2)$ in deriving the upper bound on Vtime.

By Proposition 3,

– $\mathsf{size}(t,k) = \mathsf{size}(C_{\mathsf{Batch}}) = \widetilde{O}(kR_{\mathsf{cksum}} S + |\mathcal{Q}|(\mathsf{size}(G) + S) + \mathsf{size}(C_{\downarrow})) = \widetilde{O}(kdS + k(\mathrm{poly}(d) + S) + \mathsf{size}(t, k^{\downarrow})) = \widetilde{O}(k \cdot \mathrm{poly}(d) S + \mathsf{size}(t, k^{\downarrow}))$.

– $\mathsf{depth}(t,k) = \mathsf{depth}(C_{\mathsf{Batch}}) = \widetilde{O}(1) + \widetilde{O}(G) + \mathsf{depth}(C_{\downarrow}) = \widetilde{O}(1) + \mathsf{depth}(t, k^{\downarrow})$.

– $|\langle \Psi \rangle|(t,k) = |\langle \Psi_{\mathsf{Batch}} \rangle| = |\langle \Psi_{\downarrow} \rangle| + |\langle \mathcal{Q} \rangle| + \widetilde{O}(dS) = |\langle \Psi \rangle|(t, k^{\downarrow}) + \widetilde{O}(dS + \mathrm{poly}(d))$.

To summarize,

$$\mathsf{size}(t,k) = \begin{cases} \widetilde{O}(k \cdot S) & \text{if } t < t_{\mathsf{base}},\\ \mathsf{size}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} S\lambda) & \text{if } k < k_{\mathsf{base}},\\ \mathsf{size}(t, k^{\downarrow}) + \widetilde{O}(k \cdot \mathrm{poly}(d) S) & \text{otherwise.}\end{cases}$$

$$\mathsf{depth}(t,k) = \begin{cases} \widetilde{O}(t_{\mathsf{base}}) & \text{if } t < t_{\mathsf{base}},\\ \mathsf{depth}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(1) & \text{if } k < k_{\mathsf{base}},\\ \mathsf{depth}(t, k^{\downarrow}) + \widetilde{O}(1) & \text{otherwise.}\end{cases}$$

$$\ell(t,k) = \begin{cases} \widetilde{O}(t_{\mathsf{base}}) & \text{if } t < t_{\mathsf{base}}, \\ \ell(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(1) & \text{if } k < k_{\mathsf{base}}, \\ \ell(t^{\nwarrow}, k^{\nwarrow}) + \ell(t, k^{\downarrow}) + \widetilde{O}(\mathsf{depth}(C)\log \mathsf{size}(C)) & \text{otherwise,} \end{cases}$$

$$a(t,k) = \begin{cases} \widetilde{O}(1) & \text{if } t < t_{\mathsf{base}}, \\ a(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} \cdot S\lambda) & \text{if } k < k_{\mathsf{base}}, \\ a(t^{\nwarrow}, k^{\nwarrow}) + a(t, k^{\downarrow}) + \widetilde{O}(S\lambda \mathrm{poly}(d)) & \text{otherwise,} \end{cases}$$

$$\mathsf{Ptime}(t,k) = \begin{cases} \mathrm{poly}(t_{\mathsf{base}} k S) & \text{if } t < t_{\mathsf{base}}, \\ \mathsf{Ptime}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(\mathrm{poly}(k_{\mathsf{base}} S\lambda)) & \text{if } k < k_{\mathsf{base}}, \\ \mathsf{Ptime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Ptime}(t, k^{\downarrow}) + \widetilde{O}(\mathrm{poly}(kdS\lambda \mathsf{size}(C))) & \text{otherwise,} \end{cases}$$

$$\mathsf{Vtime}(t,k) = \begin{cases} \widetilde{O}(t_{\mathsf{base}}) & \text{if } t < t_{\mathsf{base}}, \\ \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \widetilde{O}(k_{\mathsf{base}} \cdot S\lambda) & \text{if } k < k_{\mathsf{base}}, \\ \mathsf{Vtime}(t^{\nwarrow}, k^{\nwarrow}) + \mathsf{Vtime}(t, k^{\downarrow}) & \\ \quad +(\mathsf{depth}(C)\log \mathsf{size}(C) + |\langle\Phi\rangle|)S^2\lambda^2 \mathrm{poly}(d)) & \text{otherwise.} \end{cases}$$

We set the parameter $\lambda = \Lambda(T,k) = 2^{\sqrt{\log \binom{2\log T + \log k}{\log T}}}$. As analyzed below, the number of nodes in the recurrence tree is bounded by $\binom{2\log_\lambda T + \log_\lambda k}{\log_\lambda T}$, which is upper bounded by $\mathrm{poly}(\lambda)$ for our choice of λ. Solving the recurrence relations, plugging in $t_{\mathsf{base}} = \lambda$, $k_{\mathsf{base}} = d^2$, $d = \mathrm{poly}(\lambda, \sigma)$, and bounding the size of the recurrence tree by $\mathrm{poly}(\lambda)$, we have

$$\begin{aligned}
\ell(t,k) &\le \mathrm{poly}(\lambda) \cdot \widetilde{O}(\lambda) + \widetilde{O}(\mathsf{depth}(C)\log \mathsf{size}(C)) = (\mathrm{poly}(\lambda) + \mathsf{depth}(C)\log \mathsf{size}(C)) \cdot \mathrm{poly}(\sigma). \\
a(t,k) &\le \mathrm{poly}(\lambda) \cdot \mathrm{poly}(\lambda) \cdot \widetilde{O}(S) = \mathrm{poly}(\lambda) \cdot S \cdot \mathrm{poly}(\sigma). \\
\mathsf{Ptime}(t,k) &\le \mathrm{poly}(\lambda) \cdot \mathrm{poly}(T, S, \lambda, \mathsf{size}(C)) = \mathrm{poly}(S, T, \sigma, \mathsf{size}(C)). \\
\mathsf{Vtime}(t,k) &\le (\mathrm{poly}(\lambda) + \mathsf{depth}(C)\log \mathsf{size}(C) + |\langle\Phi\rangle|) \cdot \mathrm{poly}(\lambda) \cdot \widetilde{O}(S^2) \\
&= (\mathrm{poly}(\lambda) + \mathsf{depth}(C)\log \mathsf{size}(C) + |\langle\Phi\rangle|) \cdot S^2 \cdot \mathrm{poly}(\sigma). \\
\mathsf{size}(t,k) &\le (k + \mathrm{poly}(d)\lambda) \cdot S \cdot \mathrm{poly}(d) = (k + \mathrm{poly}(\lambda)) \cdot S \cdot \mathrm{poly}(\sigma). \\
\mathsf{depth}(t,k) &\le \widetilde{O}(\lambda) = \mathrm{poly}(\lambda) \cdot \mathrm{poly}(\sigma), \\
|\langle\Psi\rangle|(t,k) &\le \widetilde{O}(d^2 S\lambda + \mathrm{poly}(d)) = \mathrm{poly}(\lambda) \cdot S \cdot \mathrm{poly}(\sigma),
\end{aligned}$$

Note that $\widetilde{O}$ hides $\mathrm{polylog}(|\mathbb{F}|, n, S, T, k) = \mathrm{poly}(\sigma) \cdot \mathrm{polylog}(n, S, T, k)$ terms, and we simplify the bounds for ℓ and Vtime by Claim 4, which implies that $\mathsf{depth}(t', k')\log \mathsf{size}(t', k') \le \widetilde{O}(\lambda)$ and $|\langle\Phi\rangle|(t', k') \le \widetilde{O}(\mathrm{poly}(\lambda) \cdot S)$ for all t', k' that appear in the recurrence.

We analyze our selection of parameters $d, \lambda, t_{\mathsf{base}}, k_{\mathsf{base}}$ and show that they yield the best complexity in Section B.

5.3 Construction of Auxiliary Circuits

Proof of Proposition 2. We construct a circuit C_{base} with the given bounds that checks that $\boldsymbol{x} \in \mathcal{L}_t^k$. To do this, we can use the well-known result that any time-bounded Turing machine can be efficiently simulated by a circuit. Observe that there is a circuit of size S and constant depth (in the size of the alphabet and the number of states of $\mathcal{M}$) that, given a configuration of $\mathcal{M}$, outputs the configuration at the next time step. Call this circuit C_1. From C_1, we can construct a circuit C_t that, given a starting configuration, outputs the configuration of $\mathcal{M}$ after t time steps, by layering t copies of C_1 on top of each other. Then we can construct a circuit that checks whether $\mathcal{M}$ goes from some starting configuration to some ending configuration in t time steps by appending onto C_t a circuit that checks equality between the output of C_t and the ending configuration, which can be done using S gates and $\log S$ depth. We can do this for k t-time computations in parallel at the cost of a multiplicative factor of k in the size and an additive factor of $\log k$ in the depth.

So putting everything together, we have $\mathsf{size}(C_{\mathsf{base}}) = O(t \cdot k \cdot S)$ and $\mathsf{depth}(C_{\mathsf{base}}) = O(t + \log k + \log S)$.

□

Proof of Proposition 3. We construct a circuit C_{Batch} that satisfies the given bounds. The circuit C_{Batch} needs to (1) check that $\mathsf{cksum}_d(\boldsymbol{x}) = \chi_{\mathsf{bdry}}$, (2) expand $\langle \mathcal{Q} \rangle$ and compute $\boldsymbol{x}^{\downarrow} = \boldsymbol{x}[\mathcal{Q}, :]$, and (3) verify that $\Psi_{\downarrow}(\boldsymbol{x}^{\downarrow}) = 1$. To check (1), we compute the checksum of every one of the $2S$ columns of $\boldsymbol{x}$. By Lemma 1, computing the checksum of one column can be done with a circuit of size $O(kR_{\mathsf{cksum}})$ and depth $\widetilde{O}(1)$, and here $R_{\mathsf{cksum}} = 2d$. Therefore the $2S$ boundary-column checksums can be computed using $\widetilde{O}(kR_{\mathsf{cksum}}S)$ gates and $\widetilde{O}(1)$ depth. Then we verify equality with χ_{bdry}, which can be done using $\widetilde{O}(R_{\mathsf{cksum}}S)$ gates and $\widetilde{O}(1)$ depth. For condition (2), expanding $\langle \mathcal{Q} \rangle$ can be done using $|\mathcal{Q}|\mathsf{size}(G)$ gates and $\mathsf{depth}(G)$ depth (by taking $|\mathcal{Q}|$ parallel copies of G), and then computing $\boldsymbol{x}^{\downarrow} = \boldsymbol{x}[\mathcal{Q}, :]$ can be done using $O(|\mathcal{Q}|S)$ gates and $\widetilde{O}(1)$ depth. For condition (3), we call $C_{\downarrow}$ on $(\boldsymbol{x}^{\downarrow}, \langle \Psi_{\downarrow} \rangle)$. So in total, we have $\mathsf{size}(C_{\mathsf{Batch}}) = \widetilde{O}(kR_{\mathsf{cksum}}S + |\mathcal{Q}|(\mathsf{size}(G) + S) + \mathsf{size}(C_{\downarrow}))$ and $\mathsf{depth}(C_{\mathsf{Batch}}) = \widetilde{O}(1) + \mathsf{depth}(G) + \mathsf{depth}(C_{\downarrow})$. □

Proof of Proposition 4. We construct a circuit C_{reduce} with the given bounds that checks the following conditions.

1. Reads out $\boldsymbol{x}^{\nwarrow} = ((\boldsymbol{x}^{\nwarrow}{}_{i,j}, \boldsymbol{x}^{\nwarrow}{}_{i,j+1}))_{i,j \in [k] \times [0, \lambda - 1]} \in \{0,1\}^{2S(n) \cdot k \cdot \lambda}$ from $\boldsymbol{M}$ and verifies that $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$. To do this, we call $C_{\nwarrow}(\boldsymbol{x}^{\nwarrow}, \langle \Psi_{\nwarrow} \rangle)$. This can be done using a circuit of size $\widetilde{O}(kS\lambda) + \mathsf{size}(C_{\nwarrow})$ and depth $\widetilde{O}(1) + \mathsf{depth}(C_{\nwarrow})$.

2. Verifies that $\mathsf{cksum}_d(\boldsymbol{M}) = \chi$, where $\chi \in \mathbb{F}^{R_{\mathsf{cksum}} \times (S(\lambda+1))}$ are the hard-coded checksums. To do this, we compute the checksum of every column of $\boldsymbol{M}$ and verify that the result matches χ. By Lemma 1, computing the checksum of one column requires a circuit of size $O(kR_{\mathsf{cksum}})$ and depth $\widetilde{O}(1)$, with $R_{\mathsf{cksum}} = 2d$. Therefore, the checksums of all columns can be computed using a circuit of size $\widetilde{O}(kR_{\mathsf{cksum}}S\lambda)$ and depth $\widetilde{O}(1)$. Then we can check equality with χ using a circuit of size $\widetilde{O}(R_{\mathsf{cksum}}S\lambda)$ and depth $\widetilde{O}(1)$. So the total complexity is $\widetilde{O}(kR_{\mathsf{cksum}}S\lambda)$ gates and depth $\widetilde{O}(1)$.

3. Verifies that $\Phi(\boldsymbol{M}) = 1$. We can do this by calling C on $(\boldsymbol{M}, \langle \Phi \rangle)$.

So we have

$$\mathsf{size}(C_{\mathsf{reduce}}) = \widetilde{O}(kR_{\mathsf{cksum}}S\lambda + \mathsf{size}(C) + \mathsf{size}(C_{\nwarrow}))$$

and

$$\mathsf{depth}(C_{\mathsf{reduce}}) = \widetilde{O}(\max(\mathsf{depth}(C), \mathsf{depth}(C_{\nwarrow}))).$$

□

Proof of Proposition 5. We construct a circuit $C_{\nwarrow}$ that satisfies the given bounds. The circuit $C_{\nwarrow}$ needs to verify that (1) $\boldsymbol{x}^{\nwarrow}$'s boundary states agree with $\boldsymbol{x}$ and (2) $\Psi_{\nwarrow}(\boldsymbol{x}^{\nwarrow}) = 1$. Condition (1) is an equality check, which can be done with a circuit with $O(kS)$ gates and $\widetilde{O}(1)$ depth. To verify (2), we call $C_{\nwarrow}(\boldsymbol{x}^{\nwarrow}, \langle \Psi_{\nwarrow} \rangle)$. So in total, we have $\mathsf{size}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(kS\lambda) + \mathsf{size}(C_{\nwarrow})$ and $\mathsf{depth}(C_{\mathsf{SmallBatch}}) = \widetilde{O}(1) + \mathsf{depth}(C_{\nwarrow})$.

□

References

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23. IEEE Computer Society Press, October 1992.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, October 1992.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. In *31st FOCS*, pages 16–25. IEEE Computer Society Press, October 1990.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31. ACM Press, May 1991.

[BGHK25] Bonnie Berger, Rohan Goyal, Matthew M. Hong, and Yael Tauman Kalai. Efficiently batching unambiguous interactive proofs. In *66th FOCS*, pages 818–863. IEEE Computer Society Press, October 2025.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *20th ACM STOC*, pages 113–131. ACM Press, May 1988.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *3rd ACM STOC*, pages 151–158, May 1971.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *32nd FOCS*, pages 2–12. IEEE Computer Society Press, October 1991.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, 2015.

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

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 536–547. Springer, Berlin, Heidelberg, July 2008.

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

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[RR20] Guy N. Rothblum and Ron D. Rothblum. Batch verification and proofs of proximity with polylog overhead. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 108–138. Springer, Cham, November 2020.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient batch verification for UP. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 793–802. ACM Press, June 2013.

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

[vzGG13] Joachim von zur Gathen and Juergen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

A Interactive Proof of Proximity with Row Reduction

We show the proof of Theorem 4, which we restate here for completeness.

Theorem 4. *There exists a constant $c > 0$ such that for all σ, d, k, $L \in \mathbb{N}$, and a field $\mathbb{F}$, and any description $\langle \Phi \rangle$ of a predicate Φ with implementation circuit C whose size and depth are S and D, respectively, if the following holds:*

- $d = \Omega(\sigma \log k)$,
- $|\mathbb{F}| = \Omega(2^{\sigma} \cdot ((\sigma d L \log(|\mathbb{F}|k))^c + D \log S))$,

then there exists a $2^{-\sigma}$-sound IPP with row reduction, abbreviated by $\mathsf{IPP}(\Phi) \coloneqq (\mathsf{P}(\boldsymbol{M}), \mathsf{V})_{[\sigma, \mathbb{F}, \langle \Phi \rangle, d]}$, for the input matrix $\boldsymbol{M} \in \{0,1\}^{k \times L}$, and its output subset $\mathcal{Q} \subset [k]$ satisfies:

$$|\mathcal{Q}| \leq \left\lceil 24\sigma \cdot \frac{k}{d} \right\rceil.$$

Let $\widetilde{O}$ omit $\mathrm{polylog}(|\mathbb{F}|, k, L)$ factors. The complexity of the protocol is as follows.

- $\ell = \widetilde{O}(D \log S)$.
- $a = \widetilde{O}(dL + \mathrm{poly}(d))$.
- $\mathsf{Ptime} = \mathrm{poly}(k, L, d, S, \log |\mathbb{F}|)$.
- $\mathsf{Vtime} = \widetilde{O}(dL(D \log S + |\langle \Phi \rangle|) + \mathrm{poly}(d))$.

$|\langle \mathcal{Q} \rangle| = \widetilde{O}(\mathrm{poly}(d))$ and $|\langle \Psi \rangle| = \widetilde{O}(L + \mathrm{poly}(d))$. Let G and C be the implementation circuits of $\mathcal{Q}$ and Ψ respectively. They satisfy the following.

- $\mathsf{size}(G) = \widetilde{O}(\mathrm{poly}(d))$.
- $\mathsf{depth}(G) = \widetilde{O}(1)$.
- $\mathsf{size}(C) = \widetilde{O}(|\mathcal{Q}| \cdot L)$.
- $\mathsf{depth}(C) = \widetilde{O}(1)$.

The specific requirements on d and $|\mathbb{F}|$ are

- $d \geq 48\sigma \log k$,
- $|\mathbb{F}| \geq C_{\mathsf{GKR}} \cdot 2^{\sigma + 4c + 7}((\sigma d L \log(|\mathbb{F}|(k+1)))^c + \mathsf{depth}(C) \log \mathsf{size}(C) + 1)$,

where C_{GKR} is the constant in Theorem 3.

Organization We first set up additional preliminaries in Sections A.1 to A.3, and we present the formal protocol and its analysis in Section A.4.

A.1 Low Degree Extension and Polynomial Valuation (PVAL)

Given a subset $H \subset \mathbb{F}$ and an integer $m \in \mathbb{N}$, the *low-degree extension* (LDE) of a function $f : H^m \to \mathbb{F}$ is the unique $(|H|-1)$-individual-degree polynomial $\hat{f} : \mathbb{F}^m \to \mathbb{F}$ such that for all $\boldsymbol{s} \in H^m$, $\hat{f}(\boldsymbol{s}) = f(\boldsymbol{s})$. In this work we focus on the special case when $H = \{0,1\}$, where $\hat{f}$ is multilinear over $\mathbb{F}^m$, and is called the multilinear extension of f. Abusing the notation, given a string $x \in \mathbb{F}^n$, we first pad it with 0's such that its length becomes 2^m where $m = \lceil \log n \rceil$, and let $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ denote the multilinear extension of the function $f_x : \{0,1\}^m \to \mathbb{F}$ whose function table is x. On a sequence $\boldsymbol{j} = (\boldsymbol{j}_1, \ldots, \boldsymbol{j}_R) \in (\mathbb{F}^m)^R$ of length R, we use the shorthand $\hat{x}(\boldsymbol{j}) \coloneqq (\hat{x}(\boldsymbol{j}_1), \ldots, \hat{x}(\boldsymbol{j}_R)) \in \mathbb{F}^R$.

Consider the following *Polynomial Valuation* (PVAL) set, which is an affine subspace over $\mathbb{F}$, first defined in [RVW13].

Definition 11 (The PVAL set)**.** Let $m, R \in \mathbb{N}$. The set $\mathsf{PVAL}_{\mathbb{F}}(\boldsymbol{j}, \boldsymbol{v}) \subset \mathbb{F}^{2^m}$ is parameterized by the sequences $\boldsymbol{j} = (\boldsymbol{j}_1, \ldots, \boldsymbol{j}_R) \in (\mathbb{F}^m)^R$ and $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_R) \in \mathbb{F}^R$. It consists of all strings $x \in \mathbb{F}^{2^m}$ whose corresponding multilinear extension $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ satisfies $\hat{x}(\boldsymbol{j}) = \boldsymbol{v}$.

Note that $\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0})$ is a linear subspace of $\mathbb{F}^{2^m}$, so we can define $\Delta(\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0}))$ to be the minimum Hamming distance of a non-zero vector in $\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0})$.

A.2 The GKR protocol is Δ_c-distance-preserving

The description $\langle \Psi \rangle$ output by the GKR protocol is a pair $(\boldsymbol{j}, \boldsymbol{v}) \in \mathbb{F}^{\lceil \log n \rceil} \times \mathbb{F}$, such that $\Psi(x) = 1$ iff $\hat{x}(\boldsymbol{j}) = \boldsymbol{v}$. Regardless of P^*, $\boldsymbol{j}$ is always uniformly random and only depends on the random coins sampled by V.

Let $(\mathsf{P}_{\mathsf{GKR}}, \mathsf{V}_{\mathsf{GKR}})$ be the GKR protocol from Theorem 3. In [RVW13], the authors observed that when $(\mathsf{P}_{\mathsf{GKR}}, \mathsf{V}_{\mathsf{GKR}})$ is parallel-repeated R times for a large enough R, it is distance-preserving in the following sense: suppose the input x is d-Hamming-far from satisfying Φ, and $\boldsymbol{j} = (\boldsymbol{j}_1, \ldots, \boldsymbol{j}_R)$ and $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_R)$ are the outputs of R parallel repetitions of $(\mathsf{P}_{\mathsf{GKR}}(x), \mathsf{V}_{\mathsf{GKR}})_{[\mathbb{F}, \langle \Phi \rangle]}$, then x is d-Hamming-far from the set $\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$. Note that this increases the overall cost of the protocol by a factor of R. This generalizes to Δ_c-distance.

In the following, we naturally interpret $\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$ as a subspace of $\mathbb{F}^{k \times L}$, by treating matrices in $\mathbb{F}^{k \times L}$ as vectors in $\mathbb{F}^{kL}$. $\Delta_c(\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0}))$ is the minimum Δ_c distance of matrices in $\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0})$.

Lemma 3 (GKR is Δ_c-Distance-Preserving; Summary of Lemma 2 and Claim 3 in [BGHK25])**.** *Let $k, L, d \in \mathbb{N}$. Denote by $\mathbb{F}$ a field of characteristic 2. Consider a log-space uniform arithmetic circuit $\Phi : \mathbb{F}^{k \times L} \to \{0,1\}$ over $\mathbb{F}$, with addition and multiplication gates of fan-in 2. Let C be its implementation circuit, and let S and D be its size and depth, respectively. Let $\langle \Phi \rangle$ be the description of Φ.*

Let $\mathcal{L}_\Phi \coloneqq \left\{ \boldsymbol{M} \in \mathbb{F}^{k \times L} : \Phi(\boldsymbol{M}) = 1 \right\}$ denote the set of strings accepted by Φ. Suppose $\boldsymbol{M} \in \mathbb{F}^{k \times L}$, and that

$$\mathcal{L}_\Phi \cap \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) = \varnothing,$$

i.e. nothing in the ball $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M})$ satisfies the circuit. (See Definition 2 for the definition of $\mathcal{B}_{d,\mathbb{F}}$.)

There is a constant $C_{\mathsf{GKR}} > 0$ such that the following holds. For any prover P^* and $\sigma \in \mathbb{N}$, let

$(\boldsymbol{j}, \boldsymbol{v})$ *be the output* $\langle \mathsf{P}^*_{\mathit{GKR}}(\boldsymbol{M}), \mathsf{V}_{\mathit{GKR}} \rangle_{[\mathbb{F}, \langle \Phi \rangle]}$ *with* $R \geq 8dL \log k + \sigma + 3$ *parallel repetitions,*

$$\Pr[\mathit{PVAL}(\boldsymbol{j}, \boldsymbol{v}) \cap \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) \neq \varnothing] \leq \left(\left(C_{\mathit{GKR}} \cdot \frac{D \log S}{|\mathbb{F}|} \right)^R \cdot \left(\binom{k}{d} |\mathbb{F}|^d \right)^L \right). \tag{2}$$

$$\Pr[\Delta_c(\mathit{PVAL}(\boldsymbol{j}, \boldsymbol{0})) < 4d] \leq 2^{-\sigma-2}. \tag{3}$$

A.3 The Interactive Proof of Proximity for PVAL with Row Reduction

Lemma 4 (The Row Reduction Protocol for PVAL with Δ_c-distance, Theorem 9 in [BGHK25]; c.f. [RR20])**.** *Suppose* $k, L, d \in \mathbb{N}$*. Let* $\mathbb{F}$ *be a field. Suppose* $\sigma, R \in \mathbb{N}$*. Let* $\boldsymbol{j} = (\boldsymbol{j}_1, \ldots, \boldsymbol{j}_R) \in (\mathbb{F}^{\log k + \log L})^R$ *and* $\boldsymbol{v} = (v_1, \ldots, v_R) \in \mathbb{F}^R$*, and let* $\boldsymbol{M} \in \mathbb{F}^{k \times L}$*. For some constant* $c > 0$*, if the following conditions hold:*

- $d \geq 16\sigma \log k \in O(\sigma \log k)$,
- $|\mathbb{F}| \geq 2^{\sigma+5}(R \log k \log L)^c \in O(2^\sigma \cdot \mathrm{poly}(R, \log k, \log L))$,

then there exists an $(\ell, a, \mathsf{Ptime}, \mathsf{Vtime})$ *protocol* $(\mathsf{P}_{\mathit{RR}}(\boldsymbol{M}), \mathsf{V}_{\mathit{RR}})_{[\sigma, \mathbb{F}, d, \boldsymbol{j}, \boldsymbol{v}]}$*, which either outputs* $\perp$ *or* $(\langle \mathcal{Q} \rangle, \langle \Psi \rangle)$ *that are descriptions of a set of rows* $\mathcal{Q} \subsetneq [k]$*, and a predicate* Ψ *satisfying the following properties.*

- ***Completeness:*** *If* $\boldsymbol{M} \in \mathit{PVAL}(\boldsymbol{j}, \boldsymbol{v})$*, then* $\Pr[\Psi(\boldsymbol{M}[\mathcal{Q}, :]) = 1] = 1$.
- $2^{-\sigma}$***-Soundness:*** *Suppose* $\Delta_c(\mathit{PVAL}(\boldsymbol{j}, \boldsymbol{0})) \geq 4d$*, and* $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) \cap \mathit{PVAL}(\boldsymbol{j}, \boldsymbol{v}) = \varnothing$,[6] *then for any prover strategy* P^**,*

$$\Pr\left[\langle \mathsf{P}^*_{\mathit{RR}}(\boldsymbol{M}), \mathsf{V}_{\mathit{RR}} \rangle_{[\sigma, \mathbb{F}, d, \boldsymbol{j}, \boldsymbol{v}]} = (\langle \mathcal{Q} \rangle, \langle \Psi \rangle) \text{ s.t. } \Psi(\boldsymbol{M}[\mathcal{Q}, :]) = 1 \right] \leq 2^{-\sigma}.$$

- ***Row Reduction:*** *The subset of rows* $\mathcal{Q} \subsetneq [k]$ *has size* $|\mathcal{Q}| \leq \left\lceil 8\sigma \cdot \frac{k}{d} \right\rceil$.

With $\widetilde{O}$ *hiding* $\mathrm{polylog}(|\mathbb{F}|, k, L)$ *factors, if* $R = \widetilde{O}(dL)$*, then the complexity of the protocol is as follows.*

- $\ell = \widetilde{O}(1)$.
- $a = \widetilde{O}(L + \mathrm{poly}(d))$.
- $\mathsf{Ptime} = \mathrm{poly}(kL, R \cdot \log |\mathbb{F}|)$.
- $\mathsf{Vtime} = \widetilde{O}(L + \mathrm{poly}(d))$.

The bit-lengths are $|\langle \mathcal{Q} \rangle| = \widetilde{O}(\mathrm{poly}(d))$ *and* $|\langle \Psi \rangle| = \widetilde{O}(L + \mathrm{poly}(d))$*. Let* G *and* C *be the implementation circuits of* $\mathcal{Q}$ *and* Ψ*, respectively. Then they satisfy the following.*

- $\mathsf{size}(G) = \widetilde{O}(\mathrm{poly}(d))$.
- $\mathsf{depth}(G) = \widetilde{O}(1)$.
- $\mathsf{size}(C) = \widetilde{O}(|\mathcal{Q}| \cdot L)$.
- $\mathsf{depth}(C) = \widetilde{O}(1)$.

[6] Recall that $\mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M})$ is the set of all matrices that are Δ_c-d-close to $\boldsymbol{M}$

A.4 Proof of Theorem 4

We present the protocol in Protocol 3. It simply runs the GKR protocol followed by the RR protocol with appropriate parameters.

Protocol 3 Protocol $\mathsf{IPP} = (\mathsf{P}(\boldsymbol{M}), \mathsf{V})_{[\sigma,\mathbb{F},\langle\Phi\rangle,d]}$ for checking $\Phi(\boldsymbol{M}) = 1$.

Input Parameters: $\sigma \in \mathbb{N}$, $\mathbb{F}$ is a field, $\langle\Phi\rangle$ describes a predicate $\Phi : \{0,1\}^{k\times L} \to \{0,1\}$ whose implementation circuit has size S and depth D, $d \in \mathbb{N}$.
Input Requirement: $d \geq 48\sigma \log k$, $|\mathbb{F}| \geq C_{\mathsf{GKR}} \cdot 2^{\sigma+4c+7}((\sigma dL \log(|\mathbb{F}|(k+1)))^c + \mathsf{depth}(C)\log \mathsf{size}(C) + 1)$.
Input Matrix: $\boldsymbol{M} \in \{0,1\}^{k\times L}$.
Derived Parameters: $R = 8dL(\log k + \log|\mathbb{F}|) + \sigma + 2 = \widetilde{O}(dL)$.
Verifier Output: $\langle\mathcal{Q}\rangle, \langle\Psi\rangle$.

(1) Apply the **GKR** protocol to create the intermediate claim $\boldsymbol{M} \in \mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$.
Run $(\mathsf{P}_{\mathsf{GKR}}(\boldsymbol{M}), \mathsf{V}_{\mathsf{GKR}})_{[\mathbb{F},\langle\Phi\rangle]}$ R times in parallel and obtain $\langle\Psi\rangle = (\boldsymbol{j}^{\boldsymbol{M}}, \boldsymbol{v}^{\boldsymbol{M}})$.

(2) Apply the **RR** protocol.
Run $(\mathsf{P}_{\mathsf{RR}}(\boldsymbol{M}), \mathsf{V}_{\mathsf{RR}})_{[\sigma+2,\mathbb{F},d,\boldsymbol{j}^{\boldsymbol{M}},\boldsymbol{v}^{\boldsymbol{M}}]}$ and obtain $\langle\mathcal{Q}\rangle, \langle\Psi_{\mathsf{IPP}}\rangle$.
return $\langle\mathcal{Q}\rangle, \langle\Psi_{\mathsf{IPP}}\rangle$.

Completeness follows from the completeness of both sub-protocols. Assuming $\Phi(\boldsymbol{M}) = 1$, we have:

- The GKR protocol outputs $(\boldsymbol{j}, \boldsymbol{v})$ such that $\widehat{\boldsymbol{M}}(\boldsymbol{j}) = \boldsymbol{v}$. In other words, $\boldsymbol{M} \in \mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$.
- By the completeness of the RR protocol, given that $\boldsymbol{M} \in \mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$, we have $\Psi(\boldsymbol{M}[\mathcal{Q}, :]) = 1$.

Soundness Let $\mathcal{S} \coloneqq \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) \cap \mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{v})$. If $\mathcal{L}_\Phi \cap \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) = \varnothing$, then given our parameter settings, the following holds.

$$\begin{aligned} R &\geq 8dL(\log k + \log|\mathbb{F}|) + \sigma + 2, \\ |\mathbb{F}| &> 2^{\sigma+2} \cdot (D \log S), \end{aligned}$$

so by Lemma 3,

$$\Pr[\mathcal{S} \neq \varnothing] \leq \left(C_{\mathsf{GKR}} \cdot \frac{D\log S}{|\mathbb{F}|}\right)^R \cdot \left(\binom{k}{d}|\mathbb{F}|^d\right)^L \leq 2^{-\sigma-2}. \tag{4}$$

$$\Pr[\Delta_c(\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0})) < 4d] \leq 2^{-\sigma-2}. \tag{5}$$

Moreover, the prerequisite of Lemma 4, $d \geq 48\sigma \log k \geq 16(\sigma+2)\log k$ and $|\mathbb{F}| \geq 2^{\sigma+5}(R \log k \log L)^c$, is also satisfied by the choice of the parameters, so

$$\begin{aligned}
\Pr[\Psi(\boldsymbol{M}[\mathcal{Q},:]) = 1 \mid \mathcal{L}_\Phi \cap \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) = \varnothing] &\leq \Pr[\mathcal{S} \neq \varnothing \mid \mathcal{L}_\Phi \cap \mathcal{B}_{d,\mathbb{F}}(\boldsymbol{M}) = \varnothing] \\
&\quad + \Pr[\Psi(\boldsymbol{M}[\mathcal{Q},:]) = 1 \mid \mathcal{S} = \varnothing] \\
&\leq 2^{-\sigma-2} \\
&\quad + \Pr[\Psi_{\mathsf{IPP}}(\boldsymbol{M}^{\downarrow}) = 1 \mid \mathcal{S} = \varnothing, \Delta_c(\mathsf{PVAL}(\boldsymbol{j}^{\boldsymbol{M}}, \boldsymbol{0})) \geq 4d] \qquad (6) \\
&\quad + \Pr[\Delta_c(\mathsf{PVAL}(\boldsymbol{j}, \boldsymbol{0})) < 4d] \\
&\leq 3 \cdot 2^{-\sigma-2} < 2^{-\sigma} \qquad (7)
\end{aligned}$$

Equation (6) follows from Lemma 3, and Equation (7) follows from soundness (Lemma 4) of the RR protocol as well as Equation (3) in Lemma 3.

Complexities The overall cost is the sum of R runs of the GKR protocol and one run of the RR protocol. With $\widetilde{O}$ hiding $\mathrm{polylog}(|\mathbb{F}|, k, L)$ factors, and letting $S = \mathsf{size}(C)$, $D = \mathsf{depth}(C)$, we have

- $\ell = \widetilde{O}(D \log S) + \widetilde{O}(1) = \widetilde{O}(D \log S)$.

  Note that the round complexity of R GKR protocol runs is the same as that of one run since they are run in parallel.

- $a = \widetilde{O}(\max(R \cdot \log |\mathbb{F}|, L + \mathrm{poly}(d))) = \widetilde{O}(dL + \mathrm{poly}(d))$, given that

$$\begin{aligned}
R &= 8dL(\log k + \log |\mathbb{F}|) + \sigma + 2 = \widetilde{O}(dL + \sigma) = \widetilde{O}(dL) \\
|\mathbb{F}| &= C_{\mathsf{GKR}} \cdot 2^{\sigma+4c+7}((\sigma dL \log(|\mathbb{F}|(k+1)))^c + \mathsf{depth}(C) \log \mathsf{size}(C) + 1).
\end{aligned}$$

- $\mathsf{Ptime} = \mathrm{poly}(k, L, d, S, \log |\mathbb{F}|)$.

- $\mathsf{Vtime} = \widetilde{O}(R \cdot (D \log S + |\langle \Phi \rangle|) \cdot \log |\mathbb{F}| + L + \mathrm{poly}(d)) = \widetilde{O}(dL(D \log S + |\langle \Phi \rangle|) + \mathrm{poly}(d))$.

B Analyzing Protocol Parameters

In this section, we show that our selection of parameters $d, \lambda, t_{\mathsf{base}}, k_{\mathsf{base}}$ for Theorem 7 is optimal.

To select the optimal parameters, we focus on the recurrence relation on $\mathsf{Vtime}(t, k)$. We also assume $\mathsf{depth}(C) \log \mathsf{size}(C) = |\langle \Phi \rangle| = O(1)$, since we are most interested in $\mathcal{L}_T = \mathcal{L}_T^1[\top]$, where $\top$ is the dummy predicate that accepts all strings. Since the base cases and the intermediate terms all have to be efficient, we set all four parameters to be $\mathrm{poly}(n)$. Let $v(p, q) \coloneqq \mathsf{Vtime}(\lambda^a, \lambda^b)$, $p_{\mathsf{base}} \coloneqq \log_\lambda t_{\mathsf{base}}$, $q_{\mathsf{base}} \coloneqq \log_\lambda k_{\mathsf{base}}$. Then

$$v(p,q) = \begin{cases} \widetilde{O}(\lambda^{p_{\mathsf{base}}}) & \text{if } p < p_{\mathsf{base}}, \\ v(p-1, q+1) + \widetilde{O}(\lambda^{q_{\mathsf{base}}+1} \cdot S) & \text{if } q < q_{\mathsf{base}}, \\ v(p-1, q+1) + v(p, q - \log_\lambda \frac{d}{24\sigma}) + \widetilde{O}(S^2\lambda^2 \mathrm{poly}(d)) & \text{otherwise.} \end{cases}$$

Let $\gamma \coloneqq \frac{1}{\log_\lambda \frac{d}{24\sigma}}$. We are interested in the quantity $\mathsf{Vtime}(T, k) = v(\log_\lambda T, \log_\lambda k)$. Let us call the recurrence going from $v(p, q)$ to $v(p-1, q+1)$ a $\nwarrow$-step and the recurrence going from $v(p, q)$ to

$v(p, q - \frac{1}{\gamma})$ a $\downarrow$-step. Since the recurrence terminates only when $p < p_{\mathsf{base}}$, every path that hits the base case of the recurrence tree must use $\lceil p - p_{\mathsf{base}} \rceil$ many $\nwarrow$-steps. Since all the $\nwarrow$-steps preserve the sum $p + q$, and each $\downarrow$-step decreases the sum by γ^{-1}, the total number of $\downarrow$-steps used on any path before reaching $p < p_{\mathsf{base}}$ is at most $\gamma \cdot (\lceil p - p_{\mathsf{base}} \rceil + \lceil q - q_{\mathsf{base}} \rceil) + O(1)$. When $q < q_{\mathsf{base}}$, a $\nwarrow$ step must be taken. Therefore, the number of ways to reach the base case (*i.e.* $p < p_{\mathsf{base}}$) is upper-bounded by the number of $\{\nwarrow, \downarrow\}$-paths on the grid defined by $\{(p, q) : p \geq p_{\mathsf{base}}, q \geq 0\}$ starting from $(\log_\lambda T, \log_\lambda k)$. If we let $x = \lceil \log_\lambda T - p_{\mathsf{base}} \rceil$ and $y = \lceil \log_\lambda T + \log_\lambda k - q_{\mathsf{base}} \rceil$, the number of such paths is asymptotically upper-bounded by $\binom{x+\gamma y}{x}$. The total verifier work satisfies

$$\mathsf{Vtime}(T, k) < \binom{x + \gamma y}{x} \cdot ((k_{\mathsf{base}} + t_{\mathsf{base}} \cdot \lambda^2 S) + \mathrm{poly}(d, \lambda, S)).$$

Choosing γ For a fixed choice of $k_{\mathsf{base}}, t_{\mathsf{base}}, S$, with the observation that $d = \lambda^{\gamma^{-1}}$,

$$\mathsf{Vtime}(T, k) > \max\left(\binom{x + \gamma y}{x}, \lambda^{O(\gamma^{-1})+2}\right).$$

An optimal choice of γ balances the two terms in the max. By Stirling's approximation,

$$\log \binom{x + \gamma y}{x} \approx O\left(x \log\left(1 + \frac{\gamma y}{x}\right) + \gamma y \log\left(1 + \frac{x}{\gamma y}\right)\right)$$
$$\log(\lambda^{O(\gamma^{-1})+2}) = O((\gamma^{-1} + 2) \log \lambda).$$

Let $X \coloneqq \log T$, $Y \coloneqq \log T + \log k$. Setting the above equal gives

$$(1 + 2\gamma) \log^2 \lambda = \gamma X \log\left(1 + \frac{\gamma Y}{X}\right) + \gamma^2 Y \log\left(1 + \frac{X}{\gamma Y}\right).$$

Since $\log \mathsf{Vtime}(T, k) > (\gamma^{-1} + 2) \log \lambda$, we have

$$\begin{aligned}\log^2(\mathsf{Vtime}(T, k)) &\gtrsim \frac{(\gamma^{-1} + 2)^2}{1 + 2\gamma} \left[\gamma X \log\left(1 + \frac{\gamma Y}{X}\right) + \gamma^2 Y \log\left(1 + \frac{X}{\gamma Y}\right)\right] \\ &= (1 + 2\gamma) \left[\frac{X}{\gamma} \log\left(1 + \frac{\gamma Y}{X}\right) + Y \log\left(1 + \frac{X}{\gamma Y}\right)\right] \coloneqq f(\gamma).\end{aligned}$$

Fact 5. *$f(\gamma)$ is minimized when $\gamma = \Theta(1)$.*

Proof. If $\gamma = \omega(1)$, then

$$Y \log\left(1 + \frac{X}{\gamma Y}\right) = \Omega(1),$$

because $Y \geq X$ and the approximation $\log(1 + u) = \Theta(u)$ for $u \to 0$. Therefore, $f(\gamma) \to \infty$.

Moreover, let $g(\gamma) = \left[\frac{X}{\gamma} \log\left(1 + \frac{\gamma Y}{X}\right) + Y \log\left(1 + \frac{X}{\gamma Y}\right)\right]$. We have $f(\gamma) \geq g(\gamma)$. Computing $g'(\gamma)$,

$$\begin{aligned}g'(\gamma) &= -\frac{X}{\gamma^2} \log\left(1 + \frac{\gamma Y}{X}\right) + \frac{X}{\gamma} \frac{Y/X}{1 + \gamma Y/X} + Y \frac{-X/(\gamma^2 Y)}{1 + X/(\gamma Y)} \\ &= -\frac{X}{\gamma^2} \log\left(1 + \frac{\gamma Y}{X}\right) + \frac{XY}{\gamma X + \gamma^2 Y} - \frac{XY}{\gamma^2 Y + \gamma X} \\ &= -\frac{X}{\gamma^2} \log\left(1 + \frac{\gamma Y}{X}\right) < 0.\end{aligned}$$

Therefore, g is decreasing in γ. This implies that f is minimized when $\gamma = \Theta(1)$. □

Selecting λ Given that $\gamma = \Theta(1)$, $\binom{x+\gamma y}{x} = \binom{x+y}{x}^{\Theta(1)}$. We select a λ that balances the terms in $f(\gamma)$. Indeed, if we set $\lambda = \Lambda(T,k) = \left\lceil 2^{\sqrt{\log \binom{X+Y}{X}}} \right\rceil$, then

$$\begin{aligned} f(\gamma) = \Theta(\log^2 \lambda) &= \Theta\left(\log \binom{X+Y}{X}\right), \\ \mathsf{Vtime}(T,k) &\geq \max\left(\binom{x+\gamma y}{x}, \mathrm{poly}(\lambda^{\gamma^{-1}+2})\right) \\ &= 2^{\Theta(\sqrt{f(\gamma)})} = 2^{\Theta\left(\sqrt{\log \binom{X+Y}{X}}\right)} = \lambda^{\Theta(1)}, \end{aligned}$$

By the recurrence relation's upper bound, we also have

$$\mathsf{Vtime}(T,k) \leq \binom{x+\gamma y}{x} \cdot (k_{\mathsf{base}} + t_{\mathsf{base}} \cdot \mathrm{poly}(S,\lambda)) = \lambda^{\Theta(1)} \cdot (k_{\mathsf{base}} + t_{\mathsf{base}} \cdot \mathrm{poly}(S)).$$

Selecting k_{base}, t_{base} Given that the optimal $\lambda \geq 2^{\sqrt{\log \binom{X+Y}{X}}} = \Omega(\log T + \log k)$, the following claim can be applied.

Claim 5. *Suppose $\lambda \geq x + \gamma y \in O(\log T + \log k)$, then $\binom{x+\gamma y}{x} \cdot (k_{\mathsf{base}} + t_{\mathsf{base}} \cdot \mathrm{poly}(S,\lambda))$ is minimized when $t_{\mathsf{base}} = O(\lambda)$ and $k_{\mathsf{base}} = O(d^2)$.*

Proof Sketch. Consider the perturbation

$$\begin{cases} t_{\mathsf{base}} & \mapsto t_{\mathsf{base}} \cdot \lambda, \\ x & \mapsto x - 1. \end{cases}$$

The ratio of the upper bounds is (ignoring terms unchanged in the multiplicand)

$$\frac{\binom{x-1+\gamma y}{x-1}}{\binom{x+\gamma y}{x}} \cdot \lambda = \left(\frac{\lambda}{x+\gamma y}\right) \cdot x > 1.$$

Therefore, we must set t_{base} to be as small as possible, which is $O(\lambda)$. A similar argument holds for k_{base}. □

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il