Breaking Verifiable Delay Functions in the Random Oracle Model

Ziyi Guan
ziyi.guan@epfl.ch
EPFL

Artur Riazanov
artur.riazanov@epfl.ch
EPFL

Weiqiang Yuan
weiqiang.yuan@epfl.ch
EPFL

November 5, 2024

Abstract

This work resolves the open problem of whether verifiable delay functions (VDFs) can be constructed in the random oracle model. A VDF is a cryptographic primitive that requires a long time to compute (even with parallelization), but produces a unique output that is efficiently and publicly verifiable.

We prove that VDFs with *imperfect completeness* and *computational uniqueness* do not exist in the random oracle model. This also rules out black-box constructions of VDFs from other cryptographic primitives, such as one-way permutations and collision-resistant hash functions.

Prior to our work, Mahmoody, Smith and Wu (ICALP 2020) prove that VDFs satisfying both *perfect completeness* and *perfect uniqueness* do not exist in the random oracle model; on the other hand, Ephraim, Freitag, Komargodski, and Pass (Eurocrypt 2020) construct VDFs with *perfect completeness* and *computational uniqueness* in the random oracle model assuming the hardness of repeated squaring. Our result is optimal – we bridge the current gap between previously known impossibility results and existing constructions.

Keywords: verifiable delay functions; random oracle model; query complexity; decision trees

Contents

1 Introduction

A verifiable delay function (VDF) [BBBF18] is a cryptographic primitive that requires a long *sequential* time to compute, while the output is efficiently verifiable. More specifically, a VDF is defined by two algorithms: Eval and Verify. On input x, Eval computes an output y and a proof π in time $\mathsf{t}_{\mathsf{Eval}}$, and Verify decides whether to accept (y, π) in time $\mathsf{t}_{\mathsf{Verify}}$, where $\mathsf{t}_{\mathsf{Verify}} \ll \mathsf{t}_{\mathsf{Eval}}$. The main security requirements for VDFs are *completeness*, *computational uniqueness* and *sequentiality*. Completeness says that the solution output by Eval is accepted by Verify with high probability. Computational uniqueness says that given an input x, no adversary running in time $\mathsf{poly}(\mathsf{t}_{\mathsf{Eval}})$ can find a $y' \neq \mathsf{Eval}(x)$ and a proof π' such that (y', π') convinces the verifier. Sequentiality says that no adversary running in time smaller than $\mathsf{t}_{\mathsf{Eval}}$ (with parallel processors) can compute $y = \mathsf{Eval}(x)$.

VDFs are useful in scenarios where a delay in the computation is needed to ensure that certain operations cannot be performed too quickly. It has applications in areas such as auction protocols, proof-of-work systems, cryptographic timestamping, secure multiparty computation, and building randomness beacons ([BBBF18; BBF18; Pie19; Wes19; FMPS19; EFKP20; Sta20; HHKK23]).

Another line of work using VDFs as building blocks is proving hardness of TFNP classes. Establishing the hardness of the TFNP class PPAD [Pap94], in which finding the Nash equilibrium of a non-cooperative game is the complete problem, is a long-standing open question. [BPR15; HY17; LV20; Bit+22] discuss the similarities between constructions of hard instances in PPAD and and constructions of VDFs.

A natural question to study is whether black-box constructions of VDFs are possible from unstructured primitives, like hash functions or other symmetric primitives. The starting point would be to consider constructions in the random oracle model (ROM). [EFKP20] constructs VDFs in the ROM based on hardness of repeated squaring [RSW96]. [DGMV20] shows that *tight VDFs*, a variant whose evaluation time is very close to the sequentiality requirement, do not exist in the ROM. However, it is unclear whether the impossibility on tight VDFs extend to general VDFs. [MSW20] proves that VDFs satisfying *perfect uniqueness* (a strengthening of the computational uniqueness saying that no adversary can find a different solution) cannot be constructed in the ROM; they raise whether it is possible to rule out VDFs with computational uniqueness in the ROM as an open question.

We resolve the open question in [MSW20] and close the gap between existing constructions and known lower bounds by showing that:

Verifiable delay functions do not exist in the random oracle model.

1.1 Our results

In this paper, we focus on VDFs in the random oracle model. We measure the number of queries made by Eval and Verify to the random oracle instead of their running time. Specifically, on input x, Eval computes an output y and a proof π with query complexity at most $\mathsf{q}_{\mathsf{Eval}}$, and Verify decides whether to accept (y, π) with query complexity at most $\mathsf{q}_{\mathsf{Verify}}$, where $\mathsf{q}_{\mathsf{Verify}} \ll \mathsf{q}_{\mathsf{Eval}}$. The uniqueness and sequentiality requirements are adapted accordingly.

We propose an *equivalent* reformulation for VDFs in the ROM in terms of *decision tree algorithms*. In particular, a VDF can be viewed as a set of search problems $\{S_x \subseteq \mathrm{F} \times \mathrm{Y}\}_x$ such that for every input x, $(f, y) \in S_x$ if $\mathbb{V}_y^{(x)}(f) := \mathsf{Verify}^f(x, y)$ accepts. (We omit the proof π output by Eval here for simplicity.) We say that $\{\mathbb{V}_y^{(x)}\}_{y \in \mathrm{Y}}$ determines S_x. This reformulation enables us to use techniques in query complexity to show impossibility results regarding VDFs.

Theorem 1 (Informal)**.** *Let $S \subseteq \mathrm{F} \times \mathrm{Y}$ be a search problem determined by a family of verifiers $\{\mathbb{V}_y \colon \mathrm{F} \to \{0,1\}\}_{y \in \mathrm{Y}}$ of query complexity t. Let $\mathbb{D} \colon F \to Y$ be a T-query algorithm (with arbitrarily many rounds) that computes S correctly with high probability. Then, one of the following holds:*

- *there exists an $O(t)$-round $O(t \cdot T)$-query adversary $\mathbb{A}$ that correctly computes S with non-negligible probability; or*
- *there exists an $O(t \cdot T)$-query adversary $\mathbb{B}$ who outputs $y' \neq \mathbb{D}(f)$ such that $\mathbb{V}(f, y') = 1$ with non-negligible probability.*

Using Theorem 1, we can conclude that VDFs do not exist in the random oracle model.

Corollary 1 (Informal)**.** *Suppose* $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$ *is a VDF in the ROM. It cannot satisfy computational uniqueness and sequentiality simultaneously.*

More specifically, one can construct adversaries for VDFs from the adversaries $\mathbb{A}$ and $\mathbb{B}$ in Theorem 1. Given $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$, one of the following holds:

- there exists an $O(\mathsf{q}_{\mathsf{Verify}})$-round $O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$-query adversary that computes Eval correctly with non-negligible probability (a sequentiality breaker); or
- there exists an $O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$-query adversary who outputs $y' \neq \mathsf{Eval}(x)$ that convinces the VDF verifier with non-negligible probability (a uniqueness breaker).

We emphasize that both the uniqueness breaker and the sequentiality breaker described above run in time $\mathsf{poly}(\mathsf{t}_{\mathsf{Verify}} \cdot \mathsf{t}_{\mathsf{Eval}})$, where $\mathsf{t}_{\mathsf{Eval}}$ and $\mathsf{t}_{\mathsf{Verify}}$ represent the running time of Eval and Verify, respectively. This implies that our adversaries are *optimal* in the ROM – [EFKP20] constructs a VDF that satisfies both computational uniqueness and sequentiality in the ROM assuming the hardness of repeated squaring (the RSW assumption [RSW96]). We give a detailed explanation in Section 5.3.

Corollary 1 implies that VDFs with stronger uniqueness guarantee (e.g., perfect uniqueness) do not exist in the ROM. However, we are able to prove a quantitatively better result regarding those VDFs:

Theorem 2. *Let $S \subseteq \mathrm{F} \times \mathrm{Y}$ be a search problem determined by a family of verifiers $\{\mathbb{V}_y \colon \mathrm{F} \to \{0,1\}\}_{y \in \mathrm{Y}}$ of query complexity t such that for a negligible fraction of $f \in F$, there exists more than one $y \in Y$ where $(f, y) \in S$. Then, there exists an $O(t)$-round $O(t^2)$-query adversary $\mathbb{A}$ that computes S with non-negligible probability.*

Notice that the adversary $\mathbb{A}$ in Theorem 2 only makes $O(t^2)$ queries, while the adversary $\mathbb{A}$ in Theorem 1 uses $O(t \cdot T)$ queries. The main open question remaining is to determine whether it's possible to improve Theorem 1 quantitatively.

Open Problem 1. *Let $S \subseteq \mathrm{F} \times \mathrm{Y}$ be a search problem defined by a verifier $\mathbb{V} \colon \mathrm{F} \times \mathrm{Y} \to \{0,1\}$ with query complexity t. Let $\mathbb{D} \colon \mathrm{F} \to \mathrm{Y}$ be a T-query algorithm (with arbitrarily many rounds) that computes S correctly with high probability. Assume that for uniformly sampled $f \in \mathrm{F}$, it is computationally hard to find $y' \neq D(f)$ where $V(f, y') = 1$. Is it possible to construct a* $\mathsf{poly}(t)$*-query adversary that computes S with non-negligible probability?*

1.2 Related works

VDF and related cryptographic primitives have been studied extensively in prior works. We summarize the works that are most relevant to our results.

Verifiable delay functions. [MSW20] shows that VDFs with perfect completeness (Eval outputs an accepting solution with probability 1) and adaptive perfect uniqueness cannot exist in the ROM. Our impossibility result on VDFs with imperfect completeness and non-adaptive computational uniqueness in the ROM is more general (see Definition 3.4 and Remark 3.5). In fact, we also show a stronger claim regarding VDFs with perfect uniqueness. We postpone a detailed comparison to Section 2.2. [DGMV20] presents an in-depth study of *tight VDFs*, a variant that requires the evaluation algorithm Eval to run in time almost the same as the sequentiality requirement, and proves a negative result in the ROM. Specifically, they show that there is no VDF construction that cannot be evaluated using T rounds of queries, but can be evaluated using $T + O(T^\delta)$ (for every constant $0 < \delta < 1$) rounds of queries. Note that this does not rule out VDFs that cannot be evaluated using T rounds of queries but can be evaluated using $T + \Omega(T)$ rounds of queries. [RSS20] shows that VDFs cannot be constructed in cyclic groups of known orders. In fact, their result works for generic-group delay functions, a generalization of VDFs. [EFKP20] constructs VDFs with perfect completeness and computational uniqueness in the ROM assuming the hardness of repeated squaring. We summarize the comparison between the above-mentioned works and our results in Table 1.

	Perfect Completeness	**Imperfect Completeness**
Perfect Uniqueness	[MSW20]: ✗ (ROM) Theorem 1, Theorem 2: ✗ (ROM)	Theorem 1, Theorem 2: ✗ (ROM)
Computational Uniqueness	[DGMV20]: ✗ (ROM + tight VDF) [RSS20]:✗ (cyclic groups of known orders) [EFKP20]: ✓ (ROM + repeated squaring) Theorem 1: ✗ (ROM)	Theorem 1: ✗ (ROM)

Table 1: Comparison between prior works and our results. We use the red cross mark (✗) to indicate impossibility results and the blue check mark (✓) to indicate constructions. We specify the underlying assumption for each result in parenthesis.

Proof of sequential works. VDFs are closely related to proof of sequential works (PoSWs) [MMV13; CP18; AKKPW19; DLM19; AFGK22; AC23; Abu23]. The key difference is PoSWs do not have guarantee on the uniqueness. Our results rule out the possibilities to construct VDFs with various uniqueness guarantees in the ROM; however, it is known that PoSWs can be constructed in the ROM ([MMV13; CP18; DLM19]).

Time-lock puzzles. Time-lock puzzles ([RSW96]) are similar to VDFs as they also have the uniqueness and sequentiality guarantee. In a time-lock puzzle, a generator outputs a puzzle x and a corresponding solution y efficiently. However, computing y from x still requires large sequential time. The main difference is that time-lock puzzles require the verifier to have knowledge of a *secret key* to achieve efficient verification, while VDFs are publicly verifiable. [MMV11] rules out time-lock puzzles in the ROM.

Incrementally verifiable computations. Incrementally verifiable computation (IVC) [Val08] is a cryptographic primitive that enables efficient verification for multi-step computation. It is believed, though only partially proven [HN23; BCG24], that IVC does not exist in the ROM. [BBBF18] shows there is a black-box construction of VDFs from tight IVC (where IVC prover does not have too much overhead) for iterated sequential functions. Consequently, our results rule out tight IVC for iterated sequential functions in the ROM. However, since all hard sequential functions constructions use either the random oracle or cryptographic assumptions, our results imply that tight relativized IVC (tight IVC for which the target computation itself involves calls to the oracle) cannot be constructed in the ROM. In fact, [BCG24] proves a stronger claim: relativized IVC does not exist in the ROM, even when security holds against time-bounded (instead of just query-bounded) adversaries. We leave as an open question whether our techniques can be used to prove the impossibility of standard, non-relativized IVC in the ROM.

2 Techniques

We overview the main ideas underlying our results. In Section 2.1, we discuss our reformulation of VDFs into search problems that enables us to apply techniques developed for decision tree algorithms. In Section 2.2, we provide a simpler proof for the impossibility of VDFs with perfect uniqueness in the ROM as a warm-up. In Section 2.3, we explain how to generalize the approach in Section 2.2 to prove Theorem 2. In Section 2.4, we start with an alternative proof for the impossibility of VDFs with statistical uniqueness in the ROM and explain how to adapt it to show Theorem 1.

2.1 From VDFs to search problems

Review: VDF. A VDF in the ROM is a tuple of algorithms $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$ that works as follows: for every security parameter $\lambda \in \mathbb{N}$, let the random oracle $\mathcal{O}(\lambda)$ be the uniform distribution over the set of all functions with output length λ ($\{f\colon \{0,1\}^* \to \{0,1\}^\lambda\}$):

- The evaluation function Eval gets oracle access to a random oracle function f, receives an input $x \in \mathcal{X}$ and deterministically produces an output $y \in \mathcal{Y}$. (Note that Eval should also output a proof π, we omit it in this section for simplicity. Our formal proofs use the standard VDF definition as stated in Definition 3.2.) Eval makes at most $\mathsf{q}_{\mathsf{Eval}}$ queries to f.
- The verifier Verify gets oracle access to a random oracle function f, receives input $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and deterministically decides whether to accept or reject. Verify makes at most $\mathsf{q}_{\mathsf{Verify}}$ queries to f.

VDF is *complete* if the solution computed by Eval is accepted by Verify with high probability. For ease of discussion, we consider VDFs with perfect completeness in this section (imperfect completeness is handled carefully in Sections 4 and 5). VDF satisfies *sequentiality* if no $\mathsf{r}_{\mathsf{Adv}}$-round $\mathsf{q}_{\mathsf{Adv}}$-query ($\mathsf{r}_{\mathsf{Adv}} \ll \mathsf{q}_{\mathsf{Eval}}$ and $\mathsf{q}_{\mathsf{Adv}} = O(\mathsf{q}_{\mathsf{Eval}})$) algorithm can correctly compute Eval with non-negligible probability. Moreover, we say that VDF has *perfect uniqueness* if for every input x, Verify only accepts the output $y := \mathsf{Eval}^f(x)$; VDF has *statistical uniqueness* if for every input x, Verify accepts an alternative output $y' \neq \mathsf{Eval}^f(x)$ with negligible probability; VDF has *computational uniqueness* if for every input x and every $\mathsf{poly}(\mathsf{q}_{\mathsf{Eval}})$-query adversary Adv, Verify accepts $\mathsf{Adv}^f(x) \neq \mathsf{Eval}^f(x)$ with negligible probability. Note that the above probabilities are with respect to the choice of the random oracle function f.

Review: search problems. A search problem $S \subseteq \mathrm{F} \times \mathrm{Y}$ is defined by a family of verifiers $\{\mathbb{V}_y\colon \mathrm{F} \to \{0,1\}\}_{y \in \mathrm{Y}}$, where $(f, y) \in S$ if and only if $\mathbb{V}_y(f) = 1$. We say an algorithm $\mathbb{D}\colon \mathrm{F} \to \mathrm{Y}$ computes S if for every $f \in \mathrm{F}$, $(f, \mathbb{D}(f)) \in S$.

Reformulation of VDFs. Recall that every query algorithm can be viewed as a *decision tree*: the internal nodes of the tree represent the queries, the leaves represent the solutions, and the branching is based on the answers from the oracle to the queries.

In the ROM, the efficiency of the algorithms is measured by the number of queries they make to the random oracle. Thus, the execution of every sequential algorithm can be viewed as a decision tree. The same holds for parallel algorithms except that the internal nodes are now labeled by the set of queries instead of a single query.

Formally, fix a security parameter $\lambda \in \mathbb{N}$, for every $x \in \mathcal{X}$, we define a search problem $S_x \subseteq \mathrm{F} \times \mathrm{Y}$, where $\mathrm{F} := \{f\colon \{0,1\}^* \to \{0,1\}^\lambda\}$ and $\mathrm{Y} := \mathcal{Y}$, by a family of verifiers $\{\mathbb{V}_y^{(x)}\colon \mathrm{F} \to \{0,1\}\}_{y \in \mathrm{Y}}$ where for every $y \in \mathrm{Y}$,

$$\mathbb{V}_y^{(x)}(f) = \mathsf{Verify}^f(x, y).$$

Moreover, we define $\mathbb{D}^{(x)}\colon \mathrm{F} \to \mathrm{Y}$, which computes S_x, such that for every $f \in \mathrm{F}$,

$$\mathbb{D}^{(x)}(f) := \mathsf{Eval}^f(x).$$

The convention in query complexity is that the domain of the relations is finite, whereas the domain $\mathrm{F} = \{f\colon \{0,1\}^* \to \{0,1\}^\lambda\}$ is infinite. We observe that it is sufficient to define $S_x \subseteq [2^\lambda]^n \times \mathrm{Y}$ since there is some large constant n such that VDF depends on at most n positions of the random oracle. The total number of search problems we define is $|\mathcal{X}|$. For each search problem, we have at most $|\mathcal{Y}|$ verifiers of query complexity $\mathsf{q}_{\mathsf{Verify}}$, so each of them depends on at most $2^{\lambda \mathsf{q}_{\mathsf{Verify}}+1}$ positions in $\{0,1\}^*$. Moreover, Eval has query complexity $\mathsf{q}_{\mathsf{Eval}}$, so it depends on at most $2^{\lambda \mathsf{q}_{\mathsf{Eval}}+1}$ points in the domain of f. Thus we can bound n by $n \leq |\mathcal{X}| \left(2^{\lambda \mathsf{q}_{\mathsf{Verify}}+1}|\mathcal{Y}| + 2^{\lambda \mathsf{q}_{\mathsf{Eval}}+1}\right)$.

Hence, $\mathbb{V}_y^{(x)}$ has query complexity (with respect to the input string) $t := \mathsf{q}_{\mathsf{Verify}}$ and $\mathbb{D}^{(x)}$ has query complexity $T := \mathsf{q}_{\mathsf{Eval}}$. These search problems preserve many properties of the original VDF:

- Algorithms computing these search problems can be transformed into algorithms computing the original VDF with roughly the same complexity and success probability.
- VDFs with certain sequentiality and uniqueness properties correspond to search problems with similar properties.

2.2 Warm-up: VDFs with perfect uniqueness in the ROM

As a warm-up, we present a new proof for the impossibility of VDFs with perfect uniqueness in the ROM. Our proof is inspired by the classical algorithm witnessing that *decision tree complexity* is at most the square of *certificate complexity* for total boolean functions ([AB09]).

Since VDFs have perfect completeness and perfect uniqueness, we know that for every $x \in \mathcal{X}$ and $f \in [2^\lambda]^n$, there is a unique $y \in \mathrm{Y}$ such that $(f, y) \in S_x$. Hence, according to the search problem reformulation outlined in Section 2.1, it suffices to prove the following lemma.

Lemma 1. *Let $S_x \subseteq [2^\lambda]^n \times \mathrm{Y}$ be a search problem determined by a family of verifiers $\{\mathbb{V}_y^{(x)}\}_{y\in\mathrm{Y}}$ of query complexity t such that for every $f \in [2^\lambda]^n$, $|\{y \in \mathrm{Y} : (f, y) \in S_x\}| = 1$. Then there exists an $O(t)$-round $O(t^2)$-query adversary $\mathbb{A}^{(x)}\colon [2^\lambda]^n \to \mathrm{Y}$ that computes S_x.*

For a fixed input $x \in \mathcal{X}$, let's consider the set of all accepting leaves $\{\ell_i\}_i$ of the verifiers $\{\mathbb{V}_y^{(x)}\}_{y\in\mathrm{Y}}$. Note that each leaf ℓ_i is an element in $([2^\lambda] \cup \{\star\})^n$ such that for every $f \in [2^\lambda]^n$ that agrees with ℓ_i we have $\mathbb{V}_y^{(x)}(f) = 1$ for some $y \in \mathrm{Y}$. For ease of notation, we define the domain $\mathsf{dom}(\ell)$ for each $\ell \in ([2^\lambda] \cup \{\star\})^n$ as the set of positions that are determined:

$$\mathsf{dom}(\ell) := \{i \in [n] : \ell[i] \neq \star\}.$$

For each $\ell \in ([2^\lambda] \cup \{\star\})^n$, we define its corresponding cube $\mathsf{Cube}(\ell)$ as follows:

$$\mathsf{Cube}(\ell) := \{f \in [2^\lambda]^n : \text{for all } q \in \mathsf{dom}(\ell), f[q] = \ell[q]\}.$$

Since every $f \in [2^\lambda]^n$ has a unique solution y, $\mathsf{Cube}(\ell_i)$'s are disjoint. Hence, for every $\ell_i \neq \ell_j$, there is some $q \in \mathsf{dom}(\ell_i) \cap \mathsf{dom}(\ell_j)$ such that $\ell_i[q] \neq \ell_j[q]$. In other words, if we pick an arbitrary ℓ_i and query the given random oracle function f at all positions in $\mathsf{dom}(\ell_i)$, we "learn" at least one position for every leaf $\{\ell_i\}_i$. Since $\mathbb{V}_y^{(x)}$ makes at most $\mathsf{q}_{\mathsf{Verify}}$ queries, each leaf contains at most $\mathsf{q}_{\mathsf{Verify}}$ non-$\star$ positions. Thus, repeating the above process for $\mathsf{q}_{\mathsf{Verify}}$ times suffices for an adversary to "learn" everything to determine the solution y. Hence, we can design an adversary that always outputs the correct solution y as follows:

1. Let L be the set of all accepting leaves $\{\ell_i\}_i$ of the verifiers $\{\mathbb{V}_y^{(x)}\}_{y\in\mathrm{Y}}$.
2. Initialize $p^* := \star^n$.
3. For $i \in [\mathsf{q}_{\mathsf{Verify}}]$: Choose an arbitrary leaf ℓ in L. Query the given oracle function f at all positions in $\mathsf{dom}(\ell)$. Update p^* to record the answers of f and remove from L every leaf inconsistent with p^*.
4. Output y where $\mathbb{V}_y^{(x)}(\ell) = 1$ for every $\ell \in \mathsf{Cube}(p^*)$.

Observe that the adversary makes at most t^2 queries in t rounds. However, it is not computationally efficient since it needs to go over all accepting leaves, contrary to the adversaries we designed in the proof of Theorem 1 (see Section 5.3 for a detailed discussion).

Remark 1. *[MSW20] proves VDFs with perfect completeness and perfect uniqueness do not exist in the ROM by constructing a sequentiality adversary that makes $(2\mathsf{q}_{\mathsf{Verify}} + 1) \cdot \mathsf{q}_{\mathsf{Eval}}$ queries in $2\mathsf{q}_{\mathsf{Verify}} + 1$ rounds. Qualitatively they show a similar result as Lemma 1. However, we construct a sequentiality adversary using only $\mathsf{q}_{\mathsf{Verify}}$ rounds and $\mathsf{q}_{\mathsf{Verify}}^2$ queries. Moreover, our construction works even when VDFs have imperfect completeness (see Section 4).*

2.3 VDFs with statistical uniqueness in the ROM

In this section we explain the idea behind Theorem 2. Observe that in Section 2.2 we have proved the following claim.

Claim 1 (Folklore, see e.g. [AB09])**.** *For a given collection of* certificates $\ell_1, \ldots, \ell_N \in ([2^\lambda] \cup \{\star\})^n$ *such that* $\mathsf{Cube}(\ell_i)$ *are disjoint there exists a decision tree of depth* $O((\max_{i\in[n]} |\mathsf{dom}(\ell_i)|)^2)$ *that finds the unique certificate* ℓ_i *that agrees with the given* $f \in [2^\lambda]$.

The only difference between the statement of Lemma 1 and Claim 1 is that instead of certificates we have a collection of verifiers. It is easy to see that they are equivalent: a certificate can be checked by a verifier and a verifier can be replaced with a collection of certificates corresponding to its accepting leaves.

Theorem 2 can be seen as an approximate analogue of Claim 1.

The question of whether one can prove an approximate version of Claim 1 was first asked by Rudich in [Rud88], who derived from this (then) conjecture that one-way permutations do not exist in the random oracle model. Rudich's conjecture was resolved by Kahn, Saks, and Smyth [KSS11] using Berg-Kesten-Reimer (BKR) inequality, which was conjectured in [BK85] and proved in [Rei00].

[KSS11] taken together with [Rud88] implies that given a collection of certificates $\ell_1, \ldots, \ell_N$, if a random point in $[2^\lambda]^n$ belongs to exactly one $\mathsf{Cube}(\ell_i)$ with large enough constant probability, Claim 1 still applies, hence Theorem 2 holds. There is, however, an important technical caveat that prevents this simple logic to go through.

Ambiguity of a proof. As we have remarked, the definition that we use in this section is simplified: in the standard definition $\mathsf{Eval}^f(x)$ outputs a pair (y, π) where π is a proof that can be used by Verify. [1] In the language of certificates it means that $\mathbb{V}_y^{(x)}$ corresponds to a collection of potentially non-disjoint certificates, as opposed to certificates corresponding to accepting leaves of a single decision tree that are by definition disjoint. The uniqueness property of VDF then guarantees only that certificates *corresponding to different values* y have low total intersection, hence the result of [KSS11] does not apply.

Fortunately, following up to [KSS11], Smyth [Smy02] proved that for two certificate collections $\ell_1, \ldots, \ell_N$ and $\tau_1, \ldots, \tau_L$ such that a random point belongs to exactly one of $\bigcup_{i\in[N]} \mathsf{Cube}(\ell_i)$ and $\bigcup_{i\in[L]} \mathsf{Cube}(\tau_i)$,

[1] We emphasize that although this simplification needs to be taken care of separately in this case, it does not affect the proof of Theorem 1 (see Sections 2.4 and 5).

there exists a decision tree of depth $O(t^2)$ finding a certificate agreeing with the given $f \in [2^\lambda]^n$, where t is the largest domain size among $\ell_1, \dots, \ell_N, \tau_1, \dots, \tau_N$. This would be exactly equivalent to Theorem 2 if $|\mathrm{Y}| = 2$.

In order to prove Theorem 2 in Section 4 we generalize Smyth's result to arbitrary many certificate (cube) families. The proof follows the original one pretty closely. However, a black-box reduction to the case of two families seems elusive: a naïve attempt to partition the families in two groups, apply Smyth's theorem and continue with a smaller group introduces an error multiplier logarithmic in the number of families, which is unacceptable in our case since the initial error is constant. Moreover, it is not even clear how to give any bound on $|\mathrm{Y}|$ in terms of t.

2.4 VDFs with computational uniqueness in the ROM

We explain how to prove Theorem 1. In order to tackle VDFs with computational uniqueness, we start with a different approach to rule out VDFs with statistical uniqueness. In fact, our proof has two steps:

- Step 1: We construct an adversary that computes Eval with small sequential time if the given VDF admits statistical uniqueness;
- Step 2: We show that a modified adversary works well even when VDF only has computational uniqueness.

2.4.1 Adversary for VDFs with statistical uniqueness in the ROM

We present a proof ruling out VDFs with statistical uniqueness in the ROM. We emphasize that this proof does not give the parameter specified in Theorem 2; we present it only as an intermediate step for proving Theorem 1.

Similar to Section 2.1 we use our reformulation for a given $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$. Note that now VDF only satisfies statistical uniqueness, we don't expect our sequentiality adversary $\mathbb{A}^{(x)}$ to perfectly compute S_x anymore. Rather, we show that there exists some constant C such that for every $x \in \mathcal{X}$, there is a $O(t)$-round $O(t \cdot T)$-query adversary $\mathbb{A}^{(x)}$ that computes S_x with success probability at least $1 - C \cdot \epsilon$, where $\epsilon = \mathsf{negl}(\lambda)$ is the uniqueness error of VDF.

Our proof is inspired by [MSW20, Algorithm 1], which they use to show that VDFs with perfect uniqueness cannot be constructed in the ROM. We first explain their idea and then present how we modify it to work in our setting. ([MSW20] presents their proof in terms of VDF, we rephrase it to fit into our decision tree framework.) For each input $x \in \mathcal{X}$, [MSW20] constructs an adversary $\mathbb{A}^{(x)}$ that proceeds in $2t + 1$ rounds to compute S_x. This adversary is described in Algorithm 1.

[MSW20] observes that in each iteration, if $\mathbb{A}^{(x)}(f)$ chooses a leaf that leads to some solution other than $y := \mathbb{D}^{(x)}(f)$, it queries at least one "new" position that has also been queried by $\mathbb{V}_y^{(x)}(f)$ in this iteration. Formally, let $\ell_{\mathbb{V},f}$ be the unique accepting leaf of $\mathbb{V}_y^{(x)}$ that contains f. Let p_i^* be the value of p^* at the beginning of iteration i. Since VDF satisfies perfect uniqueness, which means that for every chosen leaf ℓ_i such that $\mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f)$, the following holds:

$$\mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) = \emptyset.$$

In other words, every time $\mathbb{A}^{(x)}(f)$ records a wrong solution, it makes progress in learning the verifier's view of f. Since $\mathbb{V}_y^{(x)}$ has query complexity at most t, at most t of the recorded solutions do not equal to y, which implies that the majority of recorded solutions gives $y = \mathbb{D}^{(x)}(f)$.

Algorithm 1 Adversary $\mathbb{A}^{(x)}$ from [MSW20].

Input: $f \in [2^\lambda]^n$
Output: $y \in \mathrm{Y} \cup \{\perp\}$

1: Let $L_1 := \{\ell_i\}_i$ be the set of leaves of $\mathbb{D}^{(x)}$.
2: Initialize $p^* := \star^n$.
3: Initialize $W := [\,]$.
4: **for** $i \in [2\mathsf{q}_{\mathsf{Verify}} + 1]$ **do**
5: Choose an arbitrary leaf ℓ_i from L_i.
6: Append $\mathbb{D}^{(x)}(\ell_i)$ to W.
7: For every $q \in \mathsf{dom}(\ell_i)$, query f at q and set $p^*[q] := f[q]$.
8: Let $L_{i+1} \subseteq L_i$ be the set of all leaves in L_i that are consistent with p^*.
9: **return** y if W contains some y that wins the majority vote; $\perp$ otherwise.

However, the above adversary cannot be directly applied in the statistical uniqueness setting: *the adversary might not make progress when it records a wrong solution.*

To be more specific, if the VDF does not have perfect uniqueness, when $\mathbb{A}^{(x)}(f)$ chooses a leaf ℓ_i that leads to some solution $y' \neq \mathbb{D}^{(x)}(f)$ in round i, it is possible that the following happens:

$$\mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) \neq \emptyset.$$

Hence, we can neither record the correct solution nor learn the verifier's view of f in this case.

The above issue can be addressed by the following two modifications to the adversary $\mathbb{A}^{(x)}$:

- In each iteration, instead of choosing an arbitrary leaf ℓ_i from L_i, we need to carefully choose a leaf that "breaks less perfect uniqueness". More specifically, we choose leaf ℓ_i such that $\mathsf{Cube}(\ell_i) \cap \mathsf{Cube}(p_i^*)$ contains fewer functions $f \in [2^\lambda]^n$ that have non-unique solutions in S_x than that in $\mathsf{Cube}(p_i^*)$ (such leaf ℓ_i exists by an averaging argument).
- Our new adversary runs in $(2+\delta)\mathsf{q}_{\mathsf{Verify}}$ rounds for some constant $\delta > 0$ instead of merely $2\mathsf{q}_{\mathsf{Verify}} + 1$ rounds.

As before, we know that there are at most t rounds i such that

$$\mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f) \text{ and } \mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) = \emptyset.$$

Moreover, from statistical uniqueness, there are at most ϵ-fraction of $f \in [2^\lambda]^n$ such that there exists some $y' \in \mathrm{Y}$ where $y' \neq \mathbb{D}^{(x)}$ and $\mathbb{V}_{y'}^{(x)} = 1$. By our specific choice of leaves in each round, in expectation, there are $(2+\delta)\mathsf{q}_{\mathsf{Verify}} \cdot \epsilon$ rounds i such that

$$\mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f) \text{ and } \mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) \neq \emptyset.$$

Hence, by Markov's inequality, $\mathbb{A}^{(x)}$ records the true solution in the majority of rounds with high probability.

2.4.2 Does computational uniqueness undermine the adversary?

We briefly discuss how the above adversary $\mathbb{A}^{(x)}$ would still succeed even when VDF satisfies only computational uniqueness. (We do need to modify $\mathbb{A}^{(x)}$ further in the formal proof, but the version outlined in Section 2.4.1 is good enough for an intuitive explanation.) The rigorous proof can be found in Section 5.

In order to better understand which part of the analysis outlined in Section 2.4.1 fails after relaxing the uniqueness guarantee, we first recall the difference in the definitions of statistical uniqueness and computational uniqueness:

- *Statistical uniqueness*: For a uniformly chosen $x \in \mathcal{X}$, there are at most ϵ-fraction of $f \in [2^\lambda]^n$ such that there exists some $y' \in \mathrm{Y}$ where $y' \neq \mathbb{D}^{(x)}$ and $\mathbb{V}^{(x)}_{y'} = 1$.
- *Computational uniqueness*: For a uniformly chosen $x \in \mathcal{X}$ and every computationally-bounded adversary $\mathbb{B}^{(x)}$, there are at most ϵ-fraction of $f \in [2^\lambda]^n$ such that $\mathbb{B}$ can find some $y' \in \mathrm{Y}$ where $y' \neq \mathbb{D}^{(x)}$ and $\mathbb{V}^{(x)}_{y'} = 1$.

According to the above definitions, for a VDF that satisfies computational uniqueness, it is possible that more than ϵ-fraction of $f \in [2^\lambda]^n$ admits multiple solutions. Hence, the previous analysis in Section 2.4.1 fails to work as we cannot directly bound the number of rounds such that $\mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f)$ and $\mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) \neq \emptyset$ anymore.

Our key observation is that from such iterations we can extract non-canonical solutions for points in the intersection of $\mathsf{Cube}(p_i^* \cup \ell_i)$ and $\mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f})$: by the choice of ℓ_i, the value of $\mathbb{D}^{(x)}$ for all these points is $\mathbb{D}^{(x)}(\ell_i)$; and by definition of $\ell_{\mathbb{V},f}$, the value $\mathbb{D}^{(x)}(f)$ is accepted by the verifier. In order to exploit this observation we devise a uniqueness adversary $\mathbb{B}^{(x)}$ "coupled" with the sequentiality adversary $\mathbb{A}^{(x)}$ in Section 2.4.1, in such a way that if there are too many rounds i where $\mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f)$ and $\mathsf{Cube}(p_i^* \cup \ell_i) \cap \mathsf{Cube}(p_i^* \cup \ell_{\mathbb{V},f}) \neq \emptyset$, $\mathbb{B}^{(x)}$ breaks computational uniqueness. Since $\mathbb{B}^{(x)}$ needs to work with non-negligible probability for a uniformly random function $f \in [2^\lambda]^n$, we have to modify the sequentiality adversary $\mathbb{A}^{(x)}$ such that the non-uniqueness witnesses are distributed uniformly.

We carefully explain how one can modify the construction of $\mathbb{A}^{(x)}$ and construct an effective uniqueness adversary $\mathbb{B}^{(x)}$ to rule out VDFs with computational uniqueness in Section 5.

3 Preliminaries

3.1 VDFs in the ROM

Definition 3.1 (The random oracle model (ROM))**.** *For every $\lambda \in \mathbb{N}$, the random oracle $\mathcal{O}(\lambda)$ is the uniform distribution over the set of all functions $f\colon \{0,1\}^* \to \{0,1\}^\lambda$.*

Definition 3.2 (Verifiable delay function (VDF) [BBBF18] in the ROM)**.** *A* **verifiable delay function** VDF **in the ROM** *is a tuple of oracle-aided algorithms* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *such that for every $\lambda \in \mathbb{N}$ and $f \in \mathcal{O}(\lambda)$, the following hold:*

- $\mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \to \mathsf{pp}$*: On input the security parameter λ and the query bound $\mathsf{q}_{\mathsf{Eval}}$, the* **deterministic** *setup algorithm* Setup *outputs the public parameters* pp*, where* pp *determines a (uniformly) samplable input space $\mathcal{X}$ and an output space $\mathcal{Y}$.*

- $\mathsf{Eval}^f(\mathsf{pp}, x) \to (y, \pi)$*: On input the public parameter* pp *and an element $x \in \mathcal{X}$, the evaluation algorithm* Eval *outputs y and a proof π, where y is generated* **deterministically** *while π can be generated in a randomized way. We sometimes ignore the output proof π and write* $\mathsf{Eval}^f(\mathsf{pp}, x) \to y$ *for simplicity.*

- $\mathsf{Verify}^f(\mathsf{pp}, x, y, \pi) \to \{0,1\}$*: On input the public parameter* pp*, and element $x \in \mathcal{X}$, a value $y \in \mathcal{Y}$, and a proof π, the* **deterministic** *verification algorithm* Verify *outputs a bit indicating whether it accepts or rejects.*

We require that Setup, Eval *and* Verify *make at most* $\mathsf{q}_{\mathsf{Setup}}$, $\mathsf{q}_{\mathsf{Eval}}$ *and* $\mathsf{q}_{\mathsf{Verify}}$ *queries, respectively, to the random oracle, where* $\mathsf{q}_{\mathsf{Setup}} = \mathsf{q}_{\mathsf{Setup}}(\lambda, \mathsf{q}_{\mathsf{Eval}})$ *and* $\mathsf{q}_{\mathsf{Verify}} = \mathsf{q}_{\mathsf{Verify}}(\lambda, \mathsf{q}_{\mathsf{Eval}})$. *In practice, we want to have VDFs where* $\mathsf{q}_{\mathsf{Setup}} \ll \mathsf{q}_{\mathsf{Eval}}$ *and* $\mathsf{q}_{\mathsf{Verify}} \ll \mathsf{q}_{\mathsf{Eval}}$.

Definition 3.3 (Completeness of VDF)**.** $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *has completeness error α if for every $\lambda \in \mathbb{N}$ and $\mathsf{q}_{\mathsf{Eval}} \in \mathbb{N}$,*

$$\Pr\left[\mathsf{Verify}^f(\mathsf{pp}, x, y, \pi) = 1 \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ (y, \pi) \leftarrow \mathsf{Eval}^f(\mathsf{pp}, x) \end{array}\right] \geq 1 - \alpha(\lambda).$$

When $\alpha = 0$, we say the VDF has **perfect completeness***.*

Definition 3.4 (Non-adaptive $(\mathsf{q}_{\mathsf{Adv}}, \epsilon)$-uniqueness of VDF)**.** *For every $\mathsf{q}_{\mathsf{Adv}}$ and ϵ,* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *satisfies $(\mathsf{q}_{\mathsf{Adv}}, \epsilon)$-uniqueness if for every $\lambda \in \mathbb{N}$, $\mathsf{q}_{\mathsf{Eval}} \in \mathbb{N}$, and $\mathsf{q}_{\mathsf{Adv}}$-query adversary* Adv*,*

$$\Pr\left[\begin{array}{l} y \neq \mathsf{Eval}^f(\mathsf{pp}, x) \\ \wedge \mathsf{Verify}^f(\mathsf{pp}, x, y, \pi) = 1 \end{array} \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ (y, \pi) \leftarrow \mathsf{Adv}^f(\mathsf{pp}, x) \end{array}\right] \leq \epsilon(\lambda).$$

We say that VDF *satisfies* **perfect uniqueness** *if $\mathsf{q}_{\mathsf{Adv}}$ is unbounded and $\epsilon(\lambda) = 0$. We say that* VDF *satisfies* **statistical uniqueness** *if $\mathsf{q}_{\mathsf{Adv}}$ is unbounded and $\epsilon(\lambda) = \mathsf{negl}(\lambda)$. We say that* VDF *satisfies* **computational uniqueness** *if $\mathsf{q}_{\mathsf{Adv}} = \mathsf{poly}(\lambda, \mathsf{q}_{\mathsf{Eval}})$ and $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.*

Remark 3.5. Note that in previous works (e.g. [BBBF18; MSW20; DGMV20]), uniqueness is defined adaptively. In other words, instead of sampling an input x uniformly at random and giving to the adversary Adv as input, they allow Adv to choose the input themselves. The adaptive uniqueness is a stronger security notion than our non-adaptive uniqueness. However, since our focus in this paper is on impossibility results, we work with non-adaptive uniqueness, which implies stronger impossibility results compared to their adaptive analogues. We sometimes write "uniqueness" instead of "non-adaptive uniqueness" for simplicity; however, we always write "adaptive uniqueness" explicitly.

Definition 3.6 (($\mathsf{r}_{\mathsf{Adv}}, \mathsf{q}_{\mathsf{Adv}}, \gamma$)-sequentiality of VDF)**.** *For every* $\mathsf{r}_{\mathsf{Adv}}$*,* $\mathsf{q}_{\mathsf{Adv}}$*, and* γ*,* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *is* $(\mathsf{r}_{\mathsf{Adv}}, \mathsf{q}_{\mathsf{Adv}}, \gamma)$*-sequential if for every* $\lambda \in \mathbb{N}$*,* $\mathsf{r}_{\mathsf{Adv}} \in \mathbb{N}$*,* $\mathsf{q}_{\mathsf{Eval}} \in \mathbb{N}$*, and* $\mathsf{r}_{\mathsf{Adv}}$*-round* $\mathsf{q}_{\mathsf{Adv}}$*-query adversary* Adv*,*

$$\Pr\left[y = \mathsf{Eval}^f(\mathsf{pp}, x) \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ y \leftarrow \mathsf{Adv}^f(\mathsf{pp}, x) \end{array} \right] \le \gamma(\lambda).$$

Remark 3.7. Note that we allow the adversary in the sequentiality definition to be parallel algorithms: it can ask multiple queries in the same round, as long as the total number of queries across rounds is upper bounded by $\mathsf{q}_{\mathsf{Adv}}$. Moreover, canonical VDF definitions (e.g. [BBBF18]) require γ to be negligible in λ, here we consider the more general definition that considers various γ.

3.2 Search problems

Definition 3.8. *A* **search problem** *is defined by a relation* $S \subseteq \mathrm{F} \times \mathrm{Y}$*. We say* S *is* **determined** *by a family of nondeterministic verifiers* $\{\mathbb{V}_{y,\pi}\}_{y \in \mathrm{Y}, \pi \in \Pi}$ *if for every* $f \in \mathrm{F}, y \in \mathrm{Y}$*,* $(f, y) \in S$ *if and only if there exists some* $\pi \in \Pi$ *such that* $\mathbb{V}_{y,\pi}(f) = 1$*. We say a search problem is* **total** *if, for every* $f \in \mathrm{F}$*, there is at least one solution* y *s.t.* $(f, y) \in S$*.*

We focus on search problems with product input space $\mathrm{F} = [M]^m$ for $M, m \in \mathbb{N}$. Given $f \in [M]^m, I \subseteq [m], p \in [M]^I$, we define $f_{I \to p} \in [M]^m$ as follows:

$$f_{I \to p}[i] := \begin{cases} p[i] & i \in I \\ f[i] & i \notin I \end{cases}.$$

Definition 3.9 (Subcube)**.** *Fix* $M, m \in \mathbb{N}$*. Let* $\mathrm{F} = [M]^m$*. We say* $\mathrm{F}' \subseteq \mathrm{F}$ *is a* **(sub)cube** *if* $\mathrm{F}' = \mathrm{F}'_1 \times \cdots \times \mathrm{F}'_n$ *for some* $\mathrm{F}'_1, \ldots, \mathrm{F}'_m \subseteq [M]$*, where* $|\mathrm{F}'_i| \in \{1, M\}$ *for each* $i \in [m]$*.*

Every query algorithm can be viewed as a ***decision tree***: the internal nodes of the tree represent the queries, the leaves represent the solutions, and the branching is based on the answers from the oracle to the queries.

A ***partial assignment*** $p \in ([M] \cup \{\star\})^m$ is a length-m string, where each entry is either fixed to be some value in $[M]$, or "undetermined" (denoted by $\star$). The ***domain of*** p is defined as $\mathsf{dom}(p) := \{i : p_i \neq \star\}$.

For ease of notation, we also identify each node p in a decision tree with a partial assignment $p \in ([M] \cup \{\star\})^m$ that records the query outcomes leading to the node p, if a position i is not queried, we set $p_i := \star$.

We say an input $f \in [M]^m$ ***is consistent with a partial assignment*** p if they agree on the domain of p, i.e. $f[i] = p[i]$ for all $i \in \mathsf{dom}(p)$. We denote by $\mathsf{Cube}(p) := \{f \in [M]^m : \forall\, i \in \mathsf{dom}(p), f[i] = p[i]\}$ the set of all inputs consistent with p.

We say that ***partial assignments*** p ***and*** q ***are consistent with each other*** if they agree on every position in the intersection of their domains, i.e. for every $i \in \mathsf{dom}(p) \cap \mathsf{dom}(q)$ we have $p[i] = q[i]$. Equivalently, $\mathsf{Cube}(p) \cap \mathsf{Cube}(q) \neq \emptyset$. We use $p \cup q$ to denote the partial assignment with domain $\mathsf{dom}(p) \cup \mathsf{dom}(q)$ that is consistent with both p and q. Note that $\mathsf{Cube}(p) \cap \mathsf{Cube}(q) = \mathsf{Cube}(p \cup q)$.

Given a distribution μ over some space F, for each $\mathrm{F}' \subseteq \mathrm{F}$, we define $\mu(\mathrm{F}') := \sum_{x \in \mathrm{F}'} \mu(x)$ as the probability of a random element sampled from μ is in F'. We use $\mathcal{U}_{\mathrm{F}}$ to denote the uniform distribution over F.

For any two partial assignments $p, q \in ([M] \cup \{\star\})^m$, we say p and q are ***independent***, denoted $p \nsim^d q$, if $\mathsf{dom}(p) \cap \mathsf{dom}(q) = \emptyset$. Otherwise, we say that p and q are ***dependent***, denoted $p \sim^d q$.

Theorem 3.10 (BKR inequality)**.** *[BK85; Rei00] Let P, Q be two collections of partial assignments over $[M]^m$. Then for every product distribution μ over $[M]^m$,*

$$\mu^{\otimes 2}\left(\bigcup_{\substack{p \in P, q \in Q \\ p \nsim^d q}} \mathsf{Cube}(p) \times \mathsf{Cube}(q) \right) \leq \mu \left(\bigcup_{\substack{p \in P \\ q \in Q}} \mathsf{Cube}(p) \cap \mathsf{Cube}(q) \right).$$

3.3 VDFs to search problems

Consider $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ with completeness error α. We present the formal reformulation of VDF in terms of search problems as described in Section 2.1.

Fix $\lambda \in \mathbb{N}$ and a large enough constant n that depends on $\mathsf{q}_{\mathsf{Setup}}, \mathsf{q}_{\mathsf{Eval}}$ and $\mathsf{q}_{\mathsf{Verify}}$. The search problems are defined below:

For every leaf $\ell \in ([2^\lambda] \cup \{\star\})^n$ of the decision tree representation of Setup:

(a) Let pp denote the label of ℓ. Deduce $\mathcal{X}$ and $\mathcal{Y}$ from pp.

(b) For every $x \in \mathcal{X}$, define the search problem $S_{\ell,x} \subseteq \mathsf{Cube}(\ell) \times \mathrm{Y}$ where $\mathrm{Y} := \mathcal{Y}$ as follows:

  i. $S_{\ell,x}$ is determined by verifiers $\mathbb{V}_{y,\pi} \colon \mathsf{Cube}(\ell) \to \{0,1\}$ of query complexity $\mathsf{q}_{\mathsf{Verify}}$ which satisfy that $\mathbb{V}_{y,\pi}(f) = \mathsf{Verify}^f(\mathsf{pp}, x, y, \pi)$.

  ii. There is an algorithm $\mathbb{D} \colon \mathsf{Cube}(\ell) \to \mathrm{Y} \times \Pi$ of query complexity $\mathsf{q}_{\mathsf{Eval}}$ which satisfies that $\mathbb{D}(f) = \mathsf{Eval}^f(\mathsf{pp}, x)$ and $\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{V}_{\mathbb{D}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq 1 - \alpha_{\ell,x}$ for some $\alpha_{\ell,x} \in [0,1]$.

Moreover, for every f, let $\ell_{\mathsf{S},f}$ denote the leaf of the decision tree representation of Setup such that $f \in \mathsf{Cube}(\ell_{\mathsf{S},f})$. It follows from Definition 3.3 that

$$\mathbb{E}\left[\alpha_{\ell_{\mathsf{S},f},x} \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \end{array} \right] \leq \alpha.$$

4 VDFs with statistical uniqueness

Theorem 4.1. *Suppose* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Verify}, \mathsf{Eval})$ *is a VDF in the ROM with completeness error* α *that satisfies statistical uniqueness with error* ϵ. *Fix* $\lambda \in \mathbb{N}$. *Let* $\mathsf{q}_{\mathsf{Setup}}$ *and* $\mathsf{q}_{\mathsf{Verify}}$ *denote the query complexity of* Setup *and* Verify*, respectively. Then for every non-zero constant* δ *such that* $\alpha + \epsilon \leq \delta \leq 10^{-3}$, VDF *does not satisfy* $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}/\delta), \mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}^2/\delta), 1 - 6\sqrt{\delta})$*-sequentiality.*

According to Section 3.3, it suffices to prove the theorem below.

Theorem 4.2 (Formal version of Theorem 2)**.** *Let* $S \subseteq [M]^m \times \mathrm{Y}$ *be a search problem determined by a family of nondeterministic verifiers* $\{\mathbb{V}_{y,\pi}\}_{y \in \mathrm{Y}, \pi \in \Pi}$ *of query complexity* t. *Let* $\delta \geq 0$ *be a parameter such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[|\{y \in \mathrm{Y} \mid (\boldsymbol{f}, y) \in S\}| \neq 1] \leq \delta$$

Then for $\delta \leq 10^{-3}$*, there exists a* t/δ*-round and* t^2/δ*-query adversary* $\mathbb{A} \colon [M]^m \to \mathrm{Y}$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[(\boldsymbol{f}, \mathbb{A}(\boldsymbol{f})) \in S] \geq 1 - 6\sqrt{\delta}.$$

Proof of Theorem 4.1 by Theorem 4.2. Observe that δ in Theorem 4.2 bounds from above the sum of the uniqueness and completeness errors, i.e., $\alpha + \epsilon \leq \delta$. Let $\mathbb{A}$ be the adversary in Theorem 4.2. We construct a VDF sequentiality adversary Adv that gets oracle access to f, runs Setup to locate the leaf ℓ_f, samples $x \leftarrow \mathcal{X}$, and executes the adversary $\mathbb{A}$ corresponding to the search problem $S_{\ell_f, x}$. If follows that Adv is a $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}/\delta)$-round $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}^2/\delta)$-query algorithm that correctly computes Eval with probability at least $1 - 6\sqrt{\delta}$. □

4.1 Proof of Theorem 4.2

Given a set $\mathrm{F}' \subseteq [M]^m$, we define $\mathsf{Uniq}(\mathrm{F}') := \{f \in \mathrm{F}' : |\{y \in \mathrm{Y} \mid (f, y) \in S\}| = 1\}$ as the set of inputs in F' which have a unique solution w.r.t. S. We describe the adversary $\mathbb{A}$ in Algorithm 2. The general structure of the algorithm follows the one described in Section 2.2. For several rounds we pick a leaf of a verifier that is consistent with the current assignment and query the input in the positions corresponding to that leaf. The crucial part of the algorithm is to choose that leaf correctly.

Algorithm 2 Sequentiality-breaking adversary $\mathbb{A}$ for statistical uniqueness.

Input: $f \in [M]^m$
Output: $y \in \mathrm{Y} \cup \{\perp\}$

```
Initialize p* := ⋆^n, y_0 := ⊥.
for i ∈ [t/δ + 1] do
    for y ∈ Y do
        if U_Cube(p*)({f' ∈ Cube(p*) : (f', y) ∈ S}) ≥ 1 − 3√δ then return y.
    if U_Cube(p*)(Uniq(Cube(p*))) < 1 − √δ/2 then return ⊥.
    if i = t/δ + 1 then return ⊥.
    Set ℓ_i := FindBestLeaf(p*), where FindBestLeaf is constructed in Algorithm 3.
    For every q ∈ dom(ℓ_i), query f at q and set p*[q] := f[q].
```

There are three cases where $\mathbb{A}$ outputs an invalid solution:

1. Algorithm 2 halts at Line 4 but returns a solution y such that $(f, y) \notin S$.
2. Algorithm 2 halts at Line 5 and returns $\perp$.
3. Algorithm 2 halts at Line 6 and returns $\perp$.

We denote the above three events by E_1, E_2, E_3 respectively. We prove that each of them happens with low probability (over random $\boldsymbol{f} \leftarrow [M]^m$).

First we bound the probability of E_1 and E_2. They do not depend on the definition of FINDBESTLEAF. Let $L_{\mathbb{A}}$ denote the set of leaves of the decision tree representation of $\mathbb{A}$. Observe that for each fixed leaf $\ell \in L_{\mathbb{A}}$, $\mathbb{A}$ behaves identically for all $f \in \mathsf{Cube}(\ell)$.

Claim 4.3. $\Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_1] \leq 3\sqrt{\delta}$.

Proof. For each leaf $\ell \in L_{\mathbb{A}}$ such that $\mathbb{A}$ terminates on Line 4, let $y(\ell)$ denote the solution that $\mathbb{A}$ outputs. We have $\Pr_{\boldsymbol{f} \sim \mathsf{Cube}(\ell)}[E_1] = \Pr_{\boldsymbol{f} \sim \mathsf{Cube}(\ell)}[(\boldsymbol{f}, y(\ell)) \notin S] \leq 3\sqrt{\delta}$, where the inequality follows from the if condition on Line 4. By averaging over all the leaves in $L_{\mathbb{A}}$, we obtain the desired claim. □

Claim 4.4. $\Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_2] \leq 2\sqrt{\delta}$.

Proof. Let $L_{\mathbb{A}}^{(2)} \subseteq L_{\mathbb{A}}$ denote the set of leaves that lead to E_2. Observe that

$$\begin{aligned}
\delta &\geq 1 - \mathcal{U}_{[M]^m}(\mathsf{Uniq}([M]^m)) \\
&\geq \sum_{\ell \in L_{\mathbb{A}}^{(2)}} \mathcal{U}_{[M]^m}(\mathsf{Cube}(\ell) \setminus \mathsf{Uniq}(\mathsf{Cube}(\ell))) \\
&\geq \frac{\sqrt{\delta}}{2} \cdot \sum_{\ell \in L_{\mathbb{A}}^{(2)}} \mathcal{U}_{[M]^m}(\mathsf{Cube}(\ell)),
\end{aligned}$$

where the last inequality follows from the if condition on Line 5. As a consequence,

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_2] = \sum_{\ell \in L_{\mathbb{A}}^{(2)}} \mathcal{U}_{[M]^m}(\mathsf{Cube}(\ell)) \leq 2\sqrt{\delta}.$$ □

Claim 4.5. $\Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_3] \leq \sqrt{\delta}$.

Given Claim 4.5 we can conclude that $\mathbb{A}$ makes t^2/δ queries in t/δ rounds and succeeds with probability at least $1 - 6\sqrt{\delta}$.

4.2 Finding the best leaf: proof of Claim 4.5

Event E_3 can be described as "$\mathbb{A}$ fails to terminate in t/δ iterations". Observe that $\mathbb{A}$ must halt in Algorithm 2 if $\mathsf{Cube}(p^*)$ is contained within $\mathsf{Cube}(\ell)$ where ℓ is an assignment corresponding to any leaf of a verifier $\mathbb{V}_{y,\pi}$. We introduce a potential function to measure how far we are from this situation. Let

$$L := \{\ell \in ([M] \cup \{\star\})^m : \ell \text{ is a leaf of } \mathbb{V}_{y,\pi} \text{ for some } y \in \mathrm{Y}, \pi \in \Pi\}.$$

Let $L_{p^*} := \{\ell \in L : \ell \text{ agrees with } p^*\}$. For $i \in [t/\delta]$ the potential function w_i is defined as

$$w_i(f) := \begin{cases} \min_{\ell \in L_{p^*} : f \in \mathsf{Cube}(\ell)} |\mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)| & f \in \mathsf{Uniq}(\mathsf{Cube}(p^*)) \\ 0 & f \notin \mathsf{Uniq}(\mathsf{Cube}(p^*)) \end{cases},$$

where p^* is the value of this variable in $\mathbb{A}(f)$ at the i-th iteration. For all f such that $\mathbb{A}(f)$ halts before invoking FINDBESTLEAF in the i-th iteration, we define $w_i(f) := 0$. The routine for FINDBESTLEAF we describe in Algorithm 3 can be informally summarized as: find ℓ such that the potential function decreases for as many values of $f \in \mathsf{Cube}(p^*)$ as possible.

Algorithm 3 Subroutine FINDBESTLEAF in Algorithm 2

Input: $p^* \in ([M] \cup \{\star\})^m$
Output: $\ell \in L_{p^*}$

1: **for** $f' \in \bigcup_{\ell \in L_{p^*}} \mathsf{Cube}(\ell \cup p^*)$ **do**
2: Set $\min \ell(f') := \arg\min_{\ell \in L_{p^*} : f' \in \mathsf{Cube}(\ell \cup p^*)} |\mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)|$.
3: **for** $\ell \in L$ **do**
4: Set $C_\ell := \{f \in \mathsf{Cube}(\ell \cup p^*) : \min \ell(f) = \ell\}$.
5: **return** $\arg\max_{\ell \in L_{p^*}} |\bigcup_{\ell' \in L_{p^*} : \ell' \sim^d \ell} C_{\ell'}|$.

For each $i \in [t/\delta]$, let P_i denote the set of all possible query outcomes p^* that are passed to FINDBESTLEAF in the i-th iteration. Fix any $p^* \in \mathsf{P}_i$. For ease of notation, we abbreviate $\mathcal{U}_{\mathsf{Cube}(p^*)}$ as $\mathcal{U}$ in the rest of the proof.

For each $\ell \in L_{p^*}$, define

$$\mathrm{F}_{p^*}(\ell) := \bigcup_{\substack{\ell' \in L_{p^*} \\ \ell' \sim^d \ell}} C_{\ell'},$$

where $C_{\ell'}$ is defined at Line 4 in Algorithm 3. We prove that

Claim 4.6. *There exists some $\hat{\ell} \in L_{p^*}$ such that $\mathcal{U}(\mathrm{F}_{p^*}(\hat{\ell})) \geq (3/2)\sqrt{\delta}$.*

We defer the proof of Claim 4.6 to Section 4.3. By Claim 4.6 we have $\mathcal{U}(\mathrm{F}_{p^*}(\ell_i)) \geq (3/2)\sqrt{\delta}$ since $\ell = \ell_i$ maximizes $\mathcal{U}(\mathrm{F}_{p^*}(\ell))$ by our choice of ℓ_i (Line 5).

Let $G(p^*) := \mathrm{F}_{p^*}(\ell_i) \cap \mathsf{Uniq}(\mathsf{Cube}(p^*))$. Observe that for all $f \in G(p^*)$, we have $w_{i+1}(f) \leq w_i(f) - 1$ since $\mathsf{dom}(\ell_i) \setminus \mathsf{dom}(p^*)$ has intersection with $\mathsf{dom}(\min \ell(f)) \setminus \mathsf{dom}(p^*)$. Moreover,

$$\mathcal{U}(G(p^*)) \geq \mathcal{U}(\mathrm{F}_{p^*}(\ell_i)) - (1 - \mathcal{U}(\mathsf{Uniq}(\mathsf{Cube}(p^*)))) \geq (3/2)\sqrt{\delta} - \sqrt{\delta}/2 \geq \sqrt{\delta}.$$

Define $H_i := \mathbb{E}_{\boldsymbol{f} \leftarrow [M]^m}[w_i(\boldsymbol{f})]$. Let $\eta_i := \sum_{p^* \in \mathsf{P}_i} \mathcal{U}_{[M]^m}(\mathsf{Cube}(p^*))$ denote the fraction of inputs that survive after the i-th iteration. Then

$$\begin{aligned} H_{i+1} - H_i &\geq \sum_{p^* \in \mathsf{P}_i} \mathcal{U}_{[M]^m}(\mathsf{Cube}(p^*)) \cdot \mathcal{U}_{\mathsf{Cube}(p^*)}(G(p^*)) \\ &\geq \sqrt{\delta} \sum_{p^* \in \mathsf{P}_i} \mathcal{U}_{[M]^m}(\mathsf{Cube}(p)) \\ &= \sqrt{\delta}\eta_i. \end{aligned}$$

Observe that $\eta_1 \geq \ldots \geq \eta_{t/\delta} \geq \Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_3]$. Moreover, $H_1 \leq t, H_{t/\delta+1} \geq 0$. Thus

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[E_3] \leq \eta_{t/\delta} \leq \delta/t \sum_{i \in [t/\delta]} \eta_i \leq \frac{(1/\sqrt{\delta}) \cdot (H_{t/\delta+1} - H_1)}{t/\delta} \leq \sqrt{\delta}.$$

4.3 Proof of Claim 4.6

Algorithm 2 in $\mathbb{A}$ guarantees that $\mathcal{U}(\{f \in \mathsf{Cube}(p^*) : (f, y) \in S\}) < 1 - 3\sqrt{\delta}$ for all $y \in \mathrm{Y}$. We can find a partition $\mathrm{Y} = \mathrm{Y}_P \sqcup \mathrm{Y}_Q$ such that

$$\max \left\{ \mathcal{U}\left(\bigcup_{p \in L^P_{p^*}} \mathsf{Cube}(p \cup p^*) \right), \mathcal{U}\left(\bigcup_{q \in L^Q_{p^*}} \mathsf{Cube}(q \cup p^*) \right) \right\} \leq 1 - 3\sqrt{\delta},$$

where $L^P_{p^*} := \{\ell \in L_{p^*} : \exists y \in \mathrm{Y}_P, \mathsf{Cube}(\ell \cup p^*) \times \{y\} \subseteq S\}$ and $L^Q_{p^*}$ is defined analogously. Denote $\Gamma := L^P_{p^*} \times L^Q_{p^*}$. Algorithm 2 in $\mathbb{A}$ guarantees that $\mathcal{U}(\mathsf{Uniq}(\mathsf{Cube}(p^*))) \geq 1 - \sqrt{\delta}/2$. We can then conclude that

$$\begin{aligned}
\mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma} \mathsf{Cube}(p \cup p^*) \times \mathsf{Cube}(q \cup p^*) \right) &\geq \mathcal{U}\left(\bigcup_{p \in L^P_{p^*}} \mathsf{Cube}(p \cup p^*) \right) \cdot \mathcal{U}\left(\bigcup_{q \in L^Q_{p^*}} \mathsf{Cube}(q \cup p^*) \right) \\
&\geq (1 - 3\sqrt{\delta}) \cdot (5/2)\sqrt{\delta},
\end{aligned}$$

where the last inequality holds since the sum of the multipliers on the left-hand side is $\mathcal{U}(\mathsf{Uniq}(\mathsf{Cube}(p^*))) \geq 1 - \sqrt{\delta}/2$ and the larger one of them is at most $1 - 3\sqrt{\delta}$. By applying Theorem 3.10, we have

$$\begin{aligned}
\mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma;\ p \nsim^d q} C_p \times C_q \right) &\leq \mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma;\ p \nsim^d q} \mathsf{Cube}(p \cup p^*) \times \mathsf{Cube}(q \cup p^*) \right) \\
\text{(by Theorem 3.10)} \quad &\leq \mathcal{U}\left(\bigcup_{(p,q) \in \Gamma;\ p \nsim^d q} \mathsf{Cube}(p \cup p^*) \cap \mathsf{Cube}(q \cup p^*) \right) \\
&\leq 1 - \mathcal{U}\left(\mathsf{Uniq}(\mathsf{Cube}(p^*)) \right) \\
&\leq \sqrt{\delta}/2.
\end{aligned}$$

With $\alpha := \mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma;\ p \sim^d q} C_p \times C_q \right)$ we then have

$$\begin{aligned}
\alpha &= \mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma} \mathsf{Cube}(p \cup p^*) \times \mathsf{Cube}(q \cup p^*) \right) - \mathcal{U}^{\otimes 2}\left(\bigcup_{(p,q) \in \Gamma;\ \mathsf{p} \nsim^d q} C_p \times C_q \right) \\
&\geq 5/2(1 - 3\sqrt{\delta})\sqrt{\delta} - \sqrt{\delta}/2 = 2\sqrt{\delta} - 7.5\delta \geq (3/2)\sqrt{\delta},
\end{aligned}$$

where the last inequality requires $\delta \leq 1/225$ which we have by the assumption. Equivalently,

$$\alpha = \sum_{p \in L^P_{p^*}} \mathcal{U}(C_p) \cdot \mathcal{U}\left(\bigcup_{q \in L^Q_{p^*};\ q \sim^d p} C_q \right) = \sum_{p \in L^P_{p^*}} \mathcal{U}(C_p) \mathrm{F}_{p^*}(p) \geq \frac{3}{2}\sqrt{\delta}.$$

Since $\sum_{p \in L^P_{p^*}} \mathcal{U}(C_p) \leq 1$, by averaging argument, there exists $\hat{\ell} \in L^P_{p^*}$ such that

$$\mathcal{U}(\mathrm{F}_{p^*}(\hat{\ell})) \geq \frac{3}{2}\sqrt{\delta}.$$

5 VDFs with computational uniqueness

We discuss VDFs with computational uniqueness in the ROM.

Theorem 5.1. *Suppose* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Verify}, \mathsf{Eval})$ *is a VDF in the ROM with completeness error* α. *Fix* $\lambda \in \mathbb{N}$. *Let* $\mathsf{q}_{\mathsf{Setup}}$, $\mathsf{q}_{\mathsf{Eval}}$ *and* $\mathsf{q}_{\mathsf{Verify}}$ *denote the query complexity of* Setup, Eval *and* Verify, *respectively. Then, for every* $\mathsf{r}_{\mathsf{Adv}} > 2\mathsf{q}_{\mathsf{Verify}}$, *there exists* $\epsilon \geq 0$ *such that* VDF *does not satisfy either* $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{r}_{\mathsf{Adv}}, \mathsf{q}_{\mathsf{Setup}} + \mathsf{r}_{\mathsf{Adv}} \cdot \mathsf{q}_{\mathsf{Eval}}, \gamma)$-*sequentiality for every* $\gamma < 1 - \frac{2\mathsf{r}_{\mathsf{Adv}}}{\mathsf{r}_{\mathsf{Adv}} - 2\mathsf{q}_{\mathsf{Verify}}} \cdot \epsilon - \alpha$ *or* $(\mathsf{q}_{\mathsf{Adv}}, \epsilon)$-*uniqueness for* $\mathsf{q}_{\mathsf{Adv}} = O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$.

We show in Appendix A that Theorem 5.1 is tight by constructing a VDF in the ROM with statistical uniqueness and weaker sequentiality.

According to Section 3.3, it suffices to prove the following theorem:

Theorem 5.2 (Formal version of Theorem 1)**.** *Let* $S \subseteq [M]^m \times \mathrm{Y}$ *be a search problem, determined by nondeterministic verifiers* $\mathbb{V}$ *of query complexity at most* t. *Let* $\mathbb{D}\colon [M]^m \to \mathrm{Y} \times \Pi$ *be an algorithm of query complexity* T *such that* $\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{V}_{\mathbb{D}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq 1 - \alpha$. *Then for every* $t' > 2t$ *there is some* $\epsilon = \epsilon(t') \geq 0$ *such that either*

1. *there exists a* t'-*round adversary* $\mathbb{A}\colon [M]^m \to \mathrm{Y}$ *of query complexity* $t'T$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{A}(\boldsymbol{f}) = \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f})] \geq 1 - \frac{2t'}{t' - 2t}\epsilon - \alpha; \text{ or}$$

2. *there exists an adversary* $\mathbb{B}\colon [M]^m \to \mathrm{Y} \times \Pi$ *of query complexity* $O(t'T)$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{B}^{\mathrm{Y}}(\boldsymbol{f}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{B}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq \epsilon,$$

where $\mathbb{D}^{\mathrm{Y}}(f)$ *(resp.* $\mathbb{B}^{\mathrm{Y}}(f)$*) is the* Y-*component of* $\mathbb{D}(f)$ *(resp.* $\mathbb{B}(f)$*).*

Proof of Theorem 5.1 by Theorem 5.2. We devise two adversaries: one for breaking sequentiality, and the other for breaking computational uniqueness as follows: First, both adversaries run Setup to locate the leaf ℓ_f and sample $x \leftarrow \mathcal{X}$. Then each adversary executes the corresponding algorithm described in Theorem 5.2 for the search problem $S_{\ell_f, x}$. It follows that for every $\mathsf{r}_{\mathsf{Adv}} > 2\mathsf{q}_{\mathsf{Verify}}$, there is some $\epsilon = \epsilon(\mathsf{r}_{\mathsf{Adv}}) \geq 0$ (by averaging over all the search problems' individual ϵ) and either

1. there exists a $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{r}_{\mathsf{Adv}})$-round $(\mathsf{r}_{\mathsf{Adv}} \cdot \mathsf{q}_{\mathsf{Eval}} + \mathsf{q}_{\mathsf{Setup}})$-query adversary Adv such that

$$\Pr\left[y = \mathsf{Eval}^f(\mathsf{pp}, x) \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ y \leftarrow \mathsf{Adv}^f(\mathsf{pp}, x) \end{array} \right] \geq 1 - \frac{2\mathsf{r}_{\mathsf{Adv}}}{\mathsf{r}_{\mathsf{Adv}} - 2\mathsf{q}_{\mathsf{Verify}}} \cdot \epsilon - \alpha; \text{ or} \tag{1}$$

2. there exists an adversary Adv of query complexity $O(\mathsf{r}_{\mathsf{Adv}} \cdot \mathsf{q}_{\mathsf{Eval}} + \mathsf{q}_{\mathsf{Setup}})$ such that

$$\Pr\left[\begin{array}{l} y \neq \mathsf{Eval}^f(\mathsf{pp}, x) \\ \wedge \mathsf{Verify}^f(\mathsf{pp}, x, y, \pi) = 1 \end{array} \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ (y, \pi) \leftarrow \mathsf{Adv}^f(\mathsf{pp}, x) \end{array} \right] \geq \epsilon. \tag{2}$$

Taking for example $\mathsf{r}_{\mathsf{Adv}} = 3\mathsf{q}_{\mathsf{Verify}}$, whatever ϵ is, either (1) is non-negligible, or (2) is non-negligible. □

5.1 The sequentiality breaker

We construct the adversary $\mathbb{A}$ below.

Algorithm 4 Adversary $\mathbb{A}$, the sequentiality breaker.

Input: $f \in [M]^m$
Output: $y \in \mathrm{Y} \cup \{\perp\}$

1: $p^* := \star^m$.
2: $K := \emptyset$.
3: **for** $r \in [t']$ **do**
4: Uniformly sample $f^* \in [M]^m$ consistent with p^*.
5: Let ℓ be the unique leaf of $\mathbb{D}$ such that $\mathsf{Cube}(\ell)$ contains f^*.
6: $K := K \uplus \{$the solution associated with $\ell\}$.
7: For every $j \in \mathsf{dom}(\ell)$ such that $p^*[j] = \star$, query f at j and update $p^*[j]$ to be the query outcome.
8: **return** the majority of solutions in K if it exists; $\perp$ otherwise.

Each iteration of $\mathbb{A}$ is a single round of queries, so $\mathbb{A}$ has t' rounds and makes at most T queries in each round, thus making at most $t'T$ queries in total.

To prove the correctness, we will first go through the execution of $\mathbb{A}$ and introduce some useful notations.

Let $\overline{\mathrm{F}} := \{f : \mathbb{V}_{\mathbb{D}(f)}(f) = 1\}$ denote the set of functions computed correctly by $\mathbb{D}$. Recall that in each iteration, we choose some leaf ℓ of $\mathbb{D}$ according to some distribution conditioned on the current partial assignment p^*. Let y denote the solution associated with ℓ. For any input $f \in \overline{\mathrm{F}}$, let $\ell_{\mathbb{V},f}$ denote the unique leaf of $\mathbb{V}_{\mathbb{D}(f)}$ such that $\mathsf{Cube}(\ell_{\mathbb{V},f})$ contains f. We classify the iterations into three types according to f, p^*, ℓ:

1. $\mathbb{D}^{\mathrm{Y}}(f) \neq y$ and $\mathsf{Cube}(p^* \cup \ell) \cap \mathsf{Cube}(p^* \cup \ell_{\mathbb{V},f}) = \emptyset$.
2. $\mathbb{D}^{\mathrm{Y}}(f) \neq y$ and $\mathsf{Cube}(p^* \cup \ell) \cap \mathsf{Cube}(p^* \cup \ell_{\mathbb{V},f}) \neq \emptyset$.
3. $\mathbb{D}^{\mathrm{Y}}(f) = y$.

Let $\boldsymbol{S}^{(1)}_{r,f}$ (resp. $\boldsymbol{S}^{(2)}_{r,f}$) be random indicator variables, which equals 1 if and only if the r-th iteration is the first type (resp. second type) for input f.

Intuitively, if both the first and the second type of iteration occur with low probability then we can prove $\mathbb{A}(f) = \mathbb{D}^{\mathrm{Y}}(f)$ with high probability by simple Markov's inequality. Now assume that $\Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f}} = 1]$ is negligible where $\boldsymbol{r}$ is uniformly sampled from $[t']$. We will prove $\Pr_{\boldsymbol{r}}[\boldsymbol{S}^{(1)}_{\boldsymbol{r},f} = 1]$ is bounded for every $f \in \overline{\mathrm{F}}$, which in turn implies $\mathbb{A}$ succeeds in simulating $\mathbb{D}$ with high probability. In Section 5.2 we show that there exists an adversary breaking the computational uniqueness condition if this assumption is false.

Lemma 5.3. *Let* $\epsilon := \Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f}} = 1]$. *Then* $\Pr_{\boldsymbol{f}}[\mathbb{A}(\boldsymbol{f}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f})] \leq \frac{2t'}{t'-2t}\epsilon + \alpha$.

Proof. We first prove that $\sum_{r=1}^{t'} \boldsymbol{S}^{(1)}_{r,f} \leq t$ with probability 1 for every $f \in \overline{\mathrm{F}}$. Consider the r-th iteration, if $\boldsymbol{S}^{(1)}_{r,f} = 1$, that is, $\mathsf{Cube}(p^* \cup \ell) \cap \mathsf{Cube}(p^* \cup \ell_{\mathbb{V},f}) = \emptyset$, then there exists some index $i \in \mathsf{dom}(\ell) \cap \mathsf{dom}(\ell_{\mathbb{V},f})$ such that $\ell[i] \neq \ell_{\mathbb{V},f}[i]$. The algorithm then queries $f[i]$ in this iteration. Thus, i will not be the inconsistent index in the later iterations. Since $|\mathsf{dom}(\ell_{\mathbb{V},f})| \leq t$, we deduce that for every f, there can be at most t iterations such that $\mathsf{Cube}(p \cup \ell) \cap \mathsf{Cube}(p \cup \ell_{\mathbb{V},f}) = \emptyset$. Hence $\sum_{1 \leq r \leq t'} \boldsymbol{S}^{(1)}_{r,f} \leq t$ with probability 1.

Now let us combine the bound for $\sum_{r=1}^{t'} \boldsymbol{S}_{r,f}^{(1)}$ with the assumption that $\Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}_{\boldsymbol{r},\boldsymbol{f}}^{(2)} = 1]$ is small. Let $\epsilon' := \frac{2t'}{t'-2t}\epsilon$. By Markov's inequality, for all but on average (over the internal randomness of $\mathbb{A}$) $(\epsilon'+\alpha)$-fraction of $f \in [M]^m$ (recall $\alpha = 1-\mathcal{U}_{[M]^m}(\overline{\mathrm{F}})$), we have $f \in \overline{\mathrm{F}}$ and $\sum_{r=1}^{t'} \boldsymbol{S}_{r,f}^{(2)} < t\epsilon/\epsilon' = t'/2 - t$. For those f, $\sum_{r=1}^{t'} \boldsymbol{S}_{r,f}^{(1)} + \boldsymbol{S}_{r,f}^{(2)} < t'/2$. Thus the majority of recorded solutions are exactly $\mathbb{D}^{\mathrm{Y}}(f)$. We conclude that the algorithm succeeds with probability at least $1-\epsilon'-\alpha$. □

5.2 The uniqueness breaker

Lemma 5.4. *Let $\boldsymbol{S}_{r,f}^{(2)}$ be defined as in the last subsection and $\epsilon := \Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}_{\boldsymbol{r},\boldsymbol{f}}^{(2)} = 1]$. Then there exists an adversary $\mathbb{B}\colon [M]^m \to (\mathrm{Y}\times\Pi)\cup\{\perp\}$ making $O(t'T)$ queries such that*

$$\Pr_{\boldsymbol{f}\leftarrow[M]^m}[\mathbb{B}^{\mathrm{Y}}(\boldsymbol{f}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{B}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq \epsilon.$$

Proof. We construct the adversary $\mathbb{B}$ in Algorithm 5.

Algorithm 5 Adversary $\mathbb{B}$, the uniqueness breaker.

Input: $f \in [M]^m$
Output: $z \in (\mathrm{Y}\times\Pi)\cup\{\perp\}$

1: Run $\mathbb{D}(f)$, let I be the set of indices queried during the execution.
2: $p^* := \star^m$.
3: **for** $r \in [t']$ **do**
4: Uniformly sample $p' \leftarrow [M]^{I\setminus\mathsf{dom}(p^*)}$.
5: $f' := f_{(I\setminus\mathsf{dom}(p^*))\to p'}$.
6: $(y,\pi) := \mathbb{D}(f')$.
7: **if** $y \neq \mathbb{D}^{\mathrm{Y}}(f) \wedge \mathbb{V}_{y,\pi}(f') = 1$ **then return** (y,π).
8: Uniformly sample $f^* \in [M]^m$ consistent with p^*.
9: Let ℓ be the unique leaf of $\mathbb{D}$ such that $\mathsf{Cube}(\ell)$ contains f^*.
10: For every $j \in \mathsf{dom}(\ell)$ such that $p^*[j] = \star$, query f at j and update $p^*[j]$ to be the query outcome.
11: **return** $\perp$

Through the execution of $\mathbb{B}$, we can define $\mathcal{D}_{\mathbb{B}}$ as the following joint distribution of $(\boldsymbol{r} \in [t'], \boldsymbol{p}^* \in ([M]\cup\{\star\})^m, \boldsymbol{f} \in [M]^m, \boldsymbol{f}' \in [M]^m)$: Sample $\boldsymbol{f} \leftarrow [M]^m, \boldsymbol{r} \leftarrow [t']$ uniformly at random. Randomly simulate the for-loop in $\mathbb{B}$ on $f = \boldsymbol{f}$ for $\boldsymbol{r}-1$ iterations. Let $\boldsymbol{p}^*$ denote the partial assignment at the start of the $\boldsymbol{r}$-th iteration and $\boldsymbol{f}'$ denote the random function f' sampled in the $\boldsymbol{r}$-th iteration of $\mathbb{B}$ (Line 5). See Fig. 1 for visualization.

To prove the lemma, it suffices to show

$$\Pr_{(\boldsymbol{r},\boldsymbol{p}^*,\boldsymbol{f},\boldsymbol{f}')\leftarrow\mathcal{D}_{\mathbb{B}}}[\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}') \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f}')}(\boldsymbol{f}) = 1] \geq \epsilon. \tag{3}$$

To this end, we give an alternative view of $\mathcal{D}_{\mathbb{B}}$ based on the execution of $\mathbb{A}$.

First, we sample $\boldsymbol{f}' \leftarrow [M]^m, \boldsymbol{r} \leftarrow [t']$ uniformly at random. Then randomly simulate the for-loop in $\mathbb{A}$ on $f = \boldsymbol{f}'$ for $\boldsymbol{r}-1$ iterations and let $\boldsymbol{p}^*$ denote the partial assignment p^* at the start of $\boldsymbol{r}$-th iteration. Recall in the $\boldsymbol{r}$-th iteration, we randomly choose some leaf ℓ of $\mathbb{D}$ conditioned on $\boldsymbol{p}^*$, and denote the solution associated with ℓ by y. Let $\boldsymbol{f}$ denote the projection of $f = \boldsymbol{f}'$ on $\mathsf{Cube}(\boldsymbol{p}^* \cup \ell)$. Formally, let $J := \mathsf{dom}(\ell)\setminus\mathsf{dom}(\boldsymbol{p}^*)$ denote the set of indices fixed by ℓ but not by $\boldsymbol{p}^*$ and we can define $\boldsymbol{f} := f_{J\to\ell_J}$.

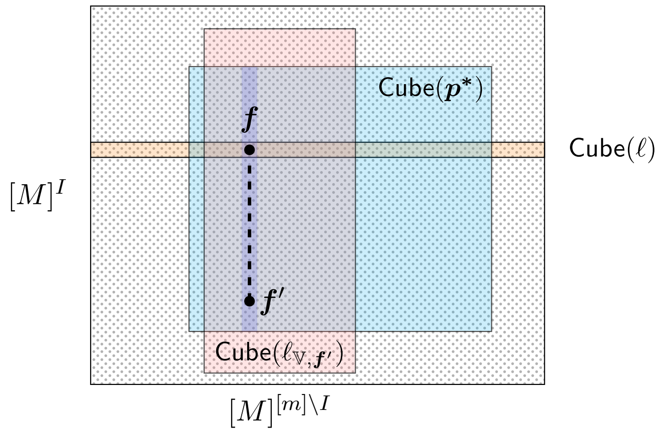

Figure 1: Distribution $\mathcal{D}_{\mathbb{B}}$.

Now observe that if $\boldsymbol{f'} \in \overline{\mathrm{F}}$ and $\boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f'}} = 1$, then $\mathsf{Cube}(\boldsymbol{p^*} \cup \ell) \cap \mathsf{Cube}(\boldsymbol{p^*} \cup \ell_{\mathbb{V},\boldsymbol{f'}}) \neq \emptyset$. Since $\boldsymbol{f} \in \mathsf{Cube}(\ell \cup \boldsymbol{p^*})$, $\boldsymbol{f}$ is consistent with $\ell_{\mathbb{V},\boldsymbol{f'}}$ on I. Moreover, $\boldsymbol{f}$ equals $\boldsymbol{f'}$ on $[m] \setminus I$, and $\mathsf{Cube}(\ell_{\mathbb{V},\boldsymbol{f'}})$ includes $\boldsymbol{f'}$, hence $\boldsymbol{f}$ is consistent with $\ell_{\mathbb{V},\boldsymbol{f'}}$ on $[m] \setminus I$. We can deduce that $\boldsymbol{f} \in \mathsf{Cube}(\ell_{\mathbb{V},\boldsymbol{f'}})$, which immediately implies $\mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f}) = 1$. Note that we also have $\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f'}) = y \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f})$ by the definition of $\boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f'}} = 1$.

Finally, let $\mathcal{D}_{\mathbb{A}}$ denote the distribution of $(\boldsymbol{r}, \boldsymbol{p^*}, \boldsymbol{f}, \boldsymbol{f'})$ according to the above sampling process. Since $\boldsymbol{f'} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f'}} = 1$ implies that $\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f'}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f}) = 1$, we can deduce that

$$\Pr_{(\boldsymbol{r},\boldsymbol{p^*},\boldsymbol{f},\boldsymbol{f'})\leftarrow\mathcal{D}_{\mathbb{A}}}[\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f'}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f}) = 1] \geq \Pr_{\boldsymbol{r},\boldsymbol{f'}}[\boldsymbol{f'} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f'}} = 1] = \epsilon.$$

Thus to prove (3), it suffices to show $\mathcal{D}_{\mathbb{B}} \equiv \mathcal{D}_{\mathbb{A}}$, that is, for every $r \in [t'], p^* \in ([M] \cup \{\star\})^m, f, f' \in [M]^m$,

$$\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{r} = r, \boldsymbol{p^*} = p^*, \boldsymbol{f} = f, \boldsymbol{f'} = f'] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{r} = r, \boldsymbol{p^*} = p^*, \boldsymbol{f} = f, \boldsymbol{f'} = f'].$$

Lemma 5.5. $\mathcal{D}_{\mathbb{A}} \equiv \mathcal{D}_{\mathbb{B}}$.

Proof. We need the following four statements.

Claim 5.6. *For every* $r \in [t']$, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{r} = r] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{r} = r]$.

Proof. Trivial since the marginal distributions of $\boldsymbol{r}$ are both uniform under $\mathbb{A}$ and $\mathbb{B}$. □

Claim 5.7. *For every* $r \in [t']$ *and* $p^* \in ([M] \cup \{\star\})^m$, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{p^*} = p^* \mid \boldsymbol{r} = r] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{p^*} = p^* \mid \boldsymbol{r} = r]$.

Proof. In both $\mathbb{A}$ and $\mathbb{B}$, $\boldsymbol{p^*}$ is the transcript of the query outcomes the following random process repeated for $r - 1$ times: Sample a uniformly random f^* consistent with the query outcome so far. Simulate $\mathbb{D}$ on f^* and query all the variables on the corresponding root-to-leaf path. □

Claim 5.8. *For every* $r \in [t'], p^* \in ([M] \cup \{\star\})^m$ *such that* $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{p^*} = p^* \mid \boldsymbol{r} = r] > 0$, *and every* $f \in [M]^m$, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r, \boldsymbol{p^*} = p^*] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r, \boldsymbol{p^*} = p^*]$.

Proof. Conditioned on $\boldsymbol{r} = r, \boldsymbol{p^*} = p^*$, it is easy to see that $\boldsymbol{f'}$ is uniformly distributed over $\mathsf{Cube}(p^*)$ under $\mathcal{D}_{\mathbb{A}}$ and $\boldsymbol{f}$ is uniformly distributed over $\mathsf{Cube}(p^*)$ under $\mathcal{D}_{\mathbb{B}}$ by Bayes' rule. It suffices to show that $\boldsymbol{f}$ is also uniformly distributed over $\mathsf{Cube}(p^*)$ under $\mathcal{D}_{\mathbb{A}}$.

Recall in the r-th round of $\mathbb{A}$, we choose some leaf ℓ of $\mathbb{D}$, and ℓ is chosen with probability $|\mathsf{Cube}(p^* \cup \ell)|/|\mathsf{Cube}(p^*)|$. Note that $\boldsymbol{f} \in \mathsf{Cube}(p^* \cup \ell)$, we only need to prove $\boldsymbol{f}$ is uniformly distributed over $\mathsf{Cube}(p^* \cup \ell)$ conditioned on ℓ is chosen. This is obvious since $\boldsymbol{f'}$ is uniformly distributed over $\mathsf{Cube}(p^*)$, and by definition, $\boldsymbol{f}$ is the projection of $\boldsymbol{f'}$ on $\mathsf{Cube}(p^* \cup \ell)$.

To conclude, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r, \boldsymbol{p^*} = p^*] = \frac{|\mathsf{Cube}(p^* \cup \ell)|}{|\mathsf{Cube}(p^*)|} \cdot \frac{1}{|\mathsf{Cube}(p^* \cup \ell)} = \frac{1}{|\mathsf{Cube}(p^*)|}$. ☐

Claim 5.9. *For every $r \in [t'], p^* \in ([M] \cup \{\star\})^m$ such that $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{p^*} = p^* \mid \boldsymbol{r} = r] > 0$, and every $f \in \mathsf{Cube}(p^*)$, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f]$.*

Proof. Without loss of generality, we assume that $f' \in \mathsf{Cube}(p^*)$, as otherwise, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] = 0$.

By Bayes' rule,

$$
\begin{aligned}
&\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] \\
&= \frac{\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*] \cdot \Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f'} = f']}{\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*]} \\
&= \Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f'} = f'].
\end{aligned}
$$

where the second equality follows since $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*] = \Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*] = \frac{1}{|\mathsf{Cube}(p^*)|}$. Let ℓ denote the unique leaf of $\mathbb{D}$ such that $f' \in \mathsf{Cube}(\ell)$ and $I = \mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)$. Now observe that

$$
\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f'} = f'] = \begin{cases} |\mathsf{Cube}(p^* \cup \ell)|/|\mathsf{Cube}(p^*)| & f_{[m]\setminus I} = f'_{[m]\setminus I} \\ 0 & \text{otherwise} \end{cases}.
$$

On the other hand, for $\mathcal{D}_{\mathbb{B}}$, given $\boldsymbol{r} = r, \boldsymbol{p^*} = p^*, \boldsymbol{f} = f$, $\boldsymbol{f'}$ uniformly from $\{f' : f'_{[m]\setminus I} = f_{[m]\setminus I}\}$. Thus $\Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] = \Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f} = f \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f'} = f'] = \Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f]$, as desired. ☐

Finally, by combining the above four claims and applying the chain rule, we deduce that $\mathcal{D}_{\mathbb{A}} \equiv \mathcal{D}_{\mathbb{B}}$. ☐

To summarize, our uniqueness breaker $\mathbb{B}$ satisfies that

$$
\begin{aligned}
\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{B}^{\mathrm{Y}}(\boldsymbol{f}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{B}(\boldsymbol{f})}(\boldsymbol{f}) = 1] &= \Pr_{(\boldsymbol{r}, \boldsymbol{p^*}, \boldsymbol{f}, \boldsymbol{f'}) \leftarrow \mathcal{D}_{\mathbb{B}}}[\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f'}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f}) = 1] \\
&= \Pr_{(\boldsymbol{r}, \boldsymbol{p^*}, \boldsymbol{f}, \boldsymbol{f'}) \leftarrow \mathcal{D}_{\mathbb{A}}}[\mathbb{D}^{\mathrm{Y}}(\boldsymbol{f'}) \neq \mathbb{D}^{\mathrm{Y}}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f}) = 1] \\
&\geq \Pr_{\boldsymbol{r}, \boldsymbol{f'}}[\boldsymbol{f'} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r}, \boldsymbol{f'}} = 1] \\
&= \epsilon.
\end{aligned}
$$

☐

5.3 Computational efficiency of the breakers

In this section we explain how to efficiently implement our sequentiality breaker $\mathbb{A}$ and uniqueness breaker $\mathbb{B}$.

Lemma 5.10. *Suppose that* Eval *is computable in time* $\mathsf{t}_{\mathsf{Eval}}$ *and* Verify *computable in time* $\mathsf{t}_{\mathsf{Verify}}$. *Then time complexity of the sequentiality adversary* $\mathbb{A}$ *(Algorithm 4) and the uniqueness adversary* $\mathbb{B}$ *(Algorithm 5) are both* $\mathsf{poly}(\mathsf{t}_{\mathsf{Verify}} \cdot \mathsf{t}_{\mathsf{Eval}})$.

Lemma 5.10 follows directly by the following implementation of the breakers (Algorithm 6 and Algorithm 7).

Algorithm 6 Uniform version of the sequentiality breaker.

Input: pp; $x \in \mathcal{X}$; oracle access to $f\colon \{0,1\}^* \to \{0,1\}^\lambda$
Output: $z \in \mathrm{Y} \cup \{\bot\}$

1: $K := \emptyset$.
2: $p^* := \emptyset$. ▷ $p^* \subseteq \{0,1\}^* \times \{0,1\}^\lambda$.
3: Define function $\mathsf{dom}(p) := \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^\lambda \colon (x,y) \in p\}$ returning the set of the first elements of a set of pairs.
4: **for** $r \in [t']$ **do**
5: Uniformly sample $f^*\colon \{0,1\}^* \to \{0,1\}^\lambda$ consistent with p^*. ▷ See Remark 5.11
6: $(y, \pi) := \mathsf{Eval}^{f^*}(\mathsf{pp}, x)$;
7: $K := K \uplus \{y\}$.
8: Let $\ell \subseteq \{0,1\}^* \times \{0,1\}^\lambda$ be the set of query-answer pairs from the execution in Line 6.
9: For every $z \in \mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)$ query $f(z)$ and update $p^* := p^* \cup (z,k)$ where $(z,k) \in \ell$.
10: **return** majority of K if it exists an $\bot$ otherwise.

Algorithm 7 Uniform version of the uniqueness breaker.

Input: pp; $x \in \mathcal{X}$; oracle access to $f\colon \{0,1\}^* \to \{0,1\}^\lambda$
Output: $z \in (\mathrm{Y} \times \Pi) \cup \{\bot\}$

1: $(y_0, \pi_0) := \mathsf{Eval}^f(\mathsf{pp}, x)$; let I be the set of random oracle queries made during the execution.
2: $p^* := \emptyset$. ▷ $p^* \subseteq \{0,1\}^* \times \{0,1\}^\lambda$.
3: Define function $\mathsf{dom}(p) := \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^\lambda \colon (x,y) \in p\}$ returning the set of the first elements of a set of pairs.
4: **for** $r \in [t']$ **do**
5: Uniformly sample $p' \leftarrow (\{0,1\}^\lambda)^{I \setminus \mathsf{dom}(p^*)}$.
6: $f' := f_{(I \setminus \mathsf{dom}(p^*)) \to p'}$. ▷ Here we only mean it symbolically, see Remark 5.11 for details.
7: $(y, \pi) := \mathsf{Eval}^f(\mathsf{pp}, x)$.
8: **if** $y \neq y_0 \wedge \mathbb{V}^{f'}(\mathsf{pp}, y, \pi) = 1$ **then return** (y, π).
9: Uniformly sample $f^*\colon \{0,1\}^* \to \{0,1\}^\lambda$ consistent with p^*. ▷ See Remark 5.11
10: Run $\mathsf{Eval}^{f^*}(\mathsf{pp}, x)$;
11: Let $\ell \subseteq \{0,1\}^* \times \{0,1\}^\lambda$ be the set of query-answer pairs from the execution in Line 10.
12: For every $z \in \mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)$ query $f(z)$ and update $p^* := p^* \cup (z,k)$ where $(z,k) \in \ell$.
13: **return** $\bot$

Remark 5.11. When we write assignments to oracles, we mean those to be defined lazily. In particular, the oracle defined in Line 6 is evaluated as follows: when $f'(z)$ is queried we first check if $z \in I \setminus \mathsf{dom}(p^*)$, if it is we return $p'(z)$, otherwise we query $f(z)$ and return the answer. The oracle defined in Line 9 is evaluated as follows: when $f^*(z)$ is queried we first check if $z \in \mathsf{dom}(p^*)$, if it is we return the unique k such that $(z, k) \in p^*$, otherwise if z was queried before we return the previously returned value, otherwise we sample k from $\{0, 1\}^\lambda$ uniformly at random and return k.

Remark 5.12. It is clear that the sequentiality breaker $\mathbb{A}$ runs in time $\mathsf{poly}(\mathsf{t}_{\mathsf{Verify}} \cdot \mathsf{t}_{\mathsf{Eval}})$. However, $\mathbb{A}$ is not parallelizable. If one can construct a sequentiality breaker that runs in parallel time smaller than $\mathsf{t}_{\mathsf{Eval}}$, it would contradict the construction in [EFKP20], which presents a VDF that satisfies computational uniqueness and sequentiality in the ROM, assuming the hardness of repeated squaring. Hence, only a polynomial improvement is possible in the time complexity in either of our breakers unless the RSW assumption [RSW96] fails.

Acknowledgments

We thank Alessandro Chiesa, Giacomo Fenzi, and Mika Göös for the helpful discussions. The authors are supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026. Ziyi Guan is partially supported by the Ethereum Foundation.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. 1st. USA: Cambridge University Press, 2009. ISBN: 0521424267.

[AC23] Hamza Abusalah and Valerio Cini. "An Incremental PoSW for General Weight Distributions". In: *Proceedings of the 42th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '23. 2023, pp. 282–311.

[AFGK22] Hamza Abusalah, Georg Fuchsbauer, Peter Gaži, and Karen Klein. "SNACKs: Leveraging Proofs of Sequential Work for Blockchain Light Clients". In: *Proceedings of the 28th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '22. 2022, pp. 806–836.

[AKKPW19] Hamza Abusalah, Chethan Kamath, Karen Klein, Krzysztof Pietrzak, and Michael Walter. "Reversible Proofs of Sequential Work". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. 2019, pp. 277–291.

[Abu23] Hamza Abusalah. *SNACKs for Proof-of-Space Blockchains*. IACR Cryptology ePrint Archive, Report 2023/806. 2023.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Function". In: *Proceedings of the 38th Annual International Cryptology Conference*. CRYPTO '18. 2018.

[BBF18] Dan Boneh, Benedikt Bünz, and Ben Fisch. *A Survey of Two Verifiable Delay Functions*. IACR Cryptology ePrint Archive, Report 2018/712. 2018.

[BCG24] Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. *Relativized Succinct Arguments in the ROM Do Not Exist*. Cryptology ePrint Archive, Paper 2024/728. 2024. URL: `https://eprint.iacr.org/2024/728`.

[BK85] J. Van Den Berg and H. Kesten. "Inequalities with applications to percolation and reliability". In: *Journal of Applied Probability* 22.3 (1985), 556–569. DOI: `10.2307/3213860`.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. "On the Cryptographic Hardness of Finding a Nash Equilibrium". In: *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '15. 2015, pp. 1480–1498.

[Bit+22] Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. "PPAD is as Hard as LWE and Iterated Squaring". In: *Proceedings of the 20th Theory of Cryptography Conference*. TCC '22. 2022, pp. 593–622.

[CP18] Bram Cohen and Krzysztof Pietrzak. "Simple Proofs of Sequential Work". In: *Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '18. 2018, pp. 451–467.

[DGMV20] Nico Döttling, Sanjam Garg, Giulio Malavolta, and Prashant Vasudevan. "Tight verifiable delay functions". In: *Proceedings of the 15th International Conference on Security and Cryptography for Networks*. SCN '20. 2020, pp. 65–84.

[DLM19] Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. "Incremental Proofs of Sequential Work". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. 2019, pp. 292–323.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. “Continuous Verifiable Delay Functions”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 125–154.

[FMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. “Verifiable Delay Functions from Supersingular Isogenies and Pairings”. In: *Proceedings of the 25th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’19. 2019, pp. 248–277.

[HHKK23] Charlotte Hoffmann, Pavel Hubáček, Chethan Kamath, and Tomáš Krňák. “(Verifiable) Delay Functions from Lucas Sequences”. In: *Proceedings of the 20th Theory of Cryptography Conference*. TCC ’23. 2023.

[HN23] Mathias Hall-Andersen and Jesper Buus Nielsen. “On Valiant’s Conjecture: Impossibility of Incrementally Verifiable Computation from Random Oracles”. In: *Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’23. 2023.

[HY17] Pavel Hubáček and Eylon Yogev. “Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds”. In: *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’17. 2017, 1352–1371.

[KSS11] Jeff Kahn, Michael Saks, and Cflifford Smyth. “The Dual BKR Inequality and Rudich’s Conjecture”. In: *Combinatorics, Probability and Computing* 20.2 (2011), 257–266. DOI: 10.1017/S0963548310000465.

[LV20] Alex Lombardi and Vinod Vaikuntanathan. “Fiat-Shamir for Repeated Squaring with Applications to PPAD-Hardness and VDFs”. In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO ’20. 2020, pp. 632–651.

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil Vadhan. “Time-Lock Puzzles in the Random Oracle Model”. In: *Proceedings of the 31st Annual International Cryptology Conference*. CRYPTO ’11. 2011, pp. 39–50.

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil Vadhan. “Publicly Verifiable Proofs of Sequential Work”. In: *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*. ITCS ’13. 2013, 373–388.

[MSW20] Mohammad Mahmoody, Caleb Smith, and David J. Wu. “Can Verifiable Delay Functions Be Based on Random Oracles?” In: *Proceedings of the 47th International Colloquium on Automata, Languages and Programming*. ICALP ’20. 2020, 83:1–83:17.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Reading, MA, USA: Addison-Wesley, 1994.

[Pie19] Krzysztof Pietrzak. “Simple Verifiable Delay Functions”. In: *Proceedings of the 10rd Innovations in Theoretical Computer Science Conference*. ITCS ’19. 2019, 60:1–60:15.

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. “Generic-Group Delay Functions Require Hidden-Order Groups”. In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 155–180. ISBN: 978-3-030-45727-3.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. *Time-Lock Puzzles and Timed-Release Crypto*. Tech. rep. 1996.

[Rei00] David Reimer. “Proof of the Van den Berg–Kesten Conjecture”. In: *Combinatorics, Probability and Computing* 9.1 (2000), 27–32. DOI: 10.1017/S0963548399004113.

[Rud88] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. Tech. rep. USA, 1988.

[Smy02] Clifford Smyth. “Reimer’s inequality and tardos’ conjecture”. In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. STOC ’02. Montreal, Quebec, Canada: Association for Computing Machinery, 2002, 218–221. ISBN: 1581134959. DOI: 10.1145/509907.509942. URL: https://doi.org/10.1145/509907.509942.

[Sta20] StarkWare Industries. *Presenting: VeeDo*. `https://medium.com/starkware/presenting-veedo-e4bbff77c7ae`. 2020.

[Val08] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *Proceedings of the 5th Theory of Cryptography Conference*. TCC '08. 2008, pp. 1–18.

[Wes19] Benjamin Wesolowski. "Efficient verifiable delay functions". In: EUROCRYPT '19 (2019), p. 623.

A Tightness of Theorem 5.1

Theorem 5.1 is essentially "tight" in terms of sequentiality: a VDF can be constructed in the ROM with statistical uniqueness and weaker sequentiality.

Lemma A.1. *Fix $\lambda \in \mathbb{N}$ and $T \in \mathbb{N}$. There exists a* $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *in which* $\mathsf{q}_{\mathsf{Setup}} = 0$*,* $\mathsf{q}_{\mathsf{Eval}} = T + 1$*, and* $\mathsf{q}_{\mathsf{Verify}} = O(1)$ *that satisfies*

– *perfect completeness,*
– $(\mathsf{q}_{\mathsf{Adv}}, \epsilon)$*-uniqueness for unbounded* $\mathsf{q}_{\mathsf{Adv}}$ *and* $\epsilon = \mathsf{negl}(\lambda)$*, and*
– $(\mathsf{r}_{\mathsf{Adv}}, \mathsf{q}'_{\mathsf{Adv}}, \gamma)$*-sequentiality for every* $\mathsf{r}_{\mathsf{Adv}} \in \mathbb{N}$*,* $\mathsf{q}'_{\mathsf{Adv}} = 2^{\lambda(T/\mathsf{r}_{\mathsf{Adv}}-1)-1}$*, and* $\gamma \geq 1 - \epsilon/4$*.*

In Theorem 5.1, we have that sequentiality error γ is upper bounded by $1 - \frac{2\mathsf{r}_{\mathsf{Adv}}}{\mathsf{r}_{\mathsf{Adv}} - 2\mathsf{q}_{\mathsf{Verify}}} \cdot \epsilon - \alpha$, which is at most $1 - 2\epsilon - \alpha$. Therefore, Lemma A.1 complements Theorem 5.1 by arguing for the existence of VDFs with perfect completeness and relaxed sequentiality error $\gamma \geq 1 - \epsilon/4$.

To show Lemma A.1, it suffices to prove the following lemma:

Lemma A.2. *For any security parameter λ, query complexity parameter $T \in \mathbb{N}^+$. Let $n = (M^T - 1)/(M - 1) + 1$. Then there is a search problem $S \subseteq [2^\lambda]^n \times [2]$ defined by verifiers $\mathbb{V}_1, \mathbb{V}_2$ and an algorithm $\mathbb{D}$ computing S which satisfies the following:*

(i) Both verifiers $\mathbb{V}_1, \mathbb{V}_2$ have query complexity $O(1)$. $\mathbb{D}$ has query complexity $T + 1$.
(ii) Exactly $1/2^\lambda$-fraction of inputs have alternative solutions, i.e. there exists $z \in \mathrm{Y}$ such that $(f, z) \in S$ but $z \neq \mathbb{D}(f)$.
(iii) For every r-round adversary $\mathbb{A}$ with query complexity at most $2^{\lambda(T/r-1)-1}$,

$$\Pr\left[\mathbb{A}(\boldsymbol{f}) = \mathbb{D}(\boldsymbol{f}) \;\middle|\; \boldsymbol{f} \leftarrow [2^\lambda]^n\right] \leq 1 - \frac{1}{2^{\lambda+2}}.$$

To construct the search problem in Lemma A.2, we define the following hard (on average) functions against parallel decision trees.

Definition A.3. *Let $M > 0$ be even, $T \in \mathbb{N}^+$. For $n \coloneqq (M^T - 1)/(M - 1)$, let $h_{M,T} \colon [M]^n \to \{0, 1\}$ be the sequential function whose computation can be defined as a complete depth-T decision tree, where different non-leaf nodes are labeled with different variables. The leaf nodes are labeled with the parity of the variable associated with their respective parent nodes so that any non-trivial subtree is balanced, namely, the subtree contains an equal number of* 0*-leaves and* 1*-leaves.*

Lemma A.4. *Any r-round algorithm computing $h_{M,T}$ with success probability $3/4$ over the uniformly random input has query complexity at least $M^{\lfloor (T-1)/r \rfloor}/2$.*

Proof. Fix $\ell = \lfloor (T-1)/r \rfloor$. We prove by induction on $R \in \mathbb{N}$ that the following alternative statement holds: Any R-round algorithm of query complexity $Q^\star \le M^l$ computing $h_{M,k\ell+1}$ has success probability at most $(1 + Q^\star/M^\ell)/2$.

When $R = 0$, any 0-round algorithm cannot make any queries. Since $h_{M,1}$ is 0 on exactly half of the inputs, the algorithm must compute $h_{M,1}$ with success probability exactly $1/2$.

Now assume that the statement is true when $R = k-1 \ge 0$. Then for $R = k$ and any k-round algorithm $\mathbb{A}_k$ of query complexity $Q^\star$ computing $h_{M,k\ell+1}$. Let $I_0 \subseteq [n(M, k\ell+1)]$ of size $|I_0| = Q_0$ denote the set of indices queried in the first round.

Recall that there is a complete depth-T decision tree computing $h_{M,k\ell+1}$, whose nodes are labeled with different variables. Let $w_1, \ldots, w_{M^\ell}$ be all the nodes on the ℓ-th level. Moreover, for any $1 \le v \le M^\ell$, let I_v be the set of indices of variables that appear in the subtree with root w_v, $g_v := h_{M,k\ell+1}|_{\mathsf{Cube}(w_v)}$. That is, g_v is a function mapping from $\mathsf{Cube}(w_v)$ to $\{0,1\}$ where $g_v(f) = h_{M,k\ell+1}(f)$ for all $f \in \mathsf{Cube}(w_v)$. Let $V := \{v : I_v \cap I_0 \ne \emptyset\}$. By the definition of $h_{M,k\ell+1}$, $I_1, \ldots, I_{M^\ell}$ are pairwise disjoint, so $|V| \le |I_0| = Q_0$.

For any $v \in [M^\ell] \setminus V$, since $\mathbb{A}_k$ does not query any variable in I_v in the first round, it performs exactly the same as some $k-1$-round $Q^\star - Q_0$-query algorithm computing g_v. It follows from the induction hypothesis and the fact that g_v is isomorphic to $h_{M,(k-1)\ell+1}$ that $\mathbb{A}_k$ computes g_v with success probability at most $(1 + (Q^\star - Q_0)/M^\ell)/2$.

Then we can bound the probability that $\mathbb{A}_k$ computes $h_{M,k\ell+1}$:

$$
\begin{aligned}
&\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{A}_k(\boldsymbol{f}) = h_{M,k\ell+1}(\boldsymbol{f})] \\
=&\frac{1}{M^\ell}\left(\sum_{v \in V} \Pr_{\boldsymbol{f} \leftarrow \mathsf{Cube}(w_v)}[\mathbb{A}_k(\boldsymbol{f}) = h_{M,k\ell+1}(\boldsymbol{f})] + \sum_{v \in [M^\ell]\setminus V} \Pr_{\boldsymbol{f} \leftarrow \mathsf{Cube}(w_v)}[\mathbb{A}_k(\boldsymbol{f}) = h_{M,k\ell+1}(\boldsymbol{f})]\right) \\
\le&\frac{1}{M^\ell}\Big(|V| + (M^\ell - |V|)(1 + (Q^\star - Q_0)/M^\ell)/2\Big) \\
\le&\frac{1}{M^\ell}\Big(Q_0 + (M^\ell - Q_0)(1 + (Q^\star - Q_0)/M^\ell)/2\Big) \\
\le&(1 + Q^\star/M^\ell)/2.
\end{aligned}
$$

Finally, by replacing $Q^\star$ with $M^\ell/2$ and observing that $t \ge r\ell + 1$, we obtain the desired claim. □

Proof of Lemma A.2. The search problem is defined by two verifiers $V_1, V_2 : [2^\lambda]^n \to \{0,1\}$: V_1 accepts all the inputs, and V_2 only accepts f such that $f_1 = 1$.

Now let us define $\mathbb{D}$. For the set of input inputs $\{f : f_1 \ne 1\}$, $\mathbb{D}$ simply outputs 1. For rest of the inputs, we embed the sequential function $h_{2^\lambda,T}$ in the subcube $\{f : f_1 = 1\}$. Specifically, we define

$$
\mathbb{D}(f) := \begin{cases} 1 & f_1 \ne 1 \\ h_{2^\lambda,T}\left(f_{[n]\setminus\{1\}}\right) + 1 & f_1 = 1 \end{cases}.
$$

It is clear that (i)(ii) hold. Note that any algorithm computing $\mathbb{D}$ with success probability at least $1 - 2^{\lambda+2}$ also computes $h_{2^\lambda,T}$ with success probability at least $3/4$. By Lemma A.4, (iii) holds. □