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

May 19, 2024

Abstract

A verifiable delay function (VDF) is a cryptographic primitive that takes a long time to compute, but produces a unique output that is efficiently and publicly verifiable.

Mahmoody, Smith and Wu (ICALP 2020) prove that VDFs satisfying both *perfect completeness* and *adaptive perfect uniqueness* do not exist in the random oracle model. Moreover, Ephraim, Freitag, Komargodski, and Pass (EUROCRYPT 2020) construct a VDF with *perfect completeness* and *computational uniqueness*, a much weaker guarantee compare to perfect uniqueness, in the random oracle model under the repeated squaring assumption.

In this work, we close the gap between existing constructions and known lower bounds by showing that VDFs with *imperfect completeness* and *non-adaptive computational uniqueness* cannot be constructed in the pure random oracle model (without additional computational assumptions).

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 takes a long *sequential* time to compute, while the output is efficiently verifiable. More specifically, a VDF contains a tuple of algorithms: Eval and Verify. On input x, Eval computes an output y in time $\mathsf{q}_{\mathsf{Eval}}$, and Verify decides whether to accept y in time $\mathsf{q}_{\mathsf{Verify}}$, where $\mathsf{q}_{\mathsf{Verify}} \ll \mathsf{q}_{\mathsf{Eval}}$. The two main security requirements for VDFs are *uniqueness* and *sequentiality*. Uniqueness says that given an input x, no (computationally bounded) adversary can find a $y' \neq \mathsf{Eval}(x)$ that convinces the verifier. Sequentiality says that no adversary running in small *sequential* time 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 potential 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.

[MSW20; DGMV20] study whether black-box constructions of VDFs are possible from unstructured primitives, like hash functions or other symmetric primitives. Their starting point is to consider constructions in the random oracle model (ROM). [MSW20] proves that VDFs satisfying *perfect uniqueness* (no adversary can find a different solution) cannot be constructed in the ROM. [DGMV20] shows that *tight VDFs*, where the evaluation time is close to the sequentiality requirement, do not exist in the ROM. On the other hand, [Pie19] constructs a VDF with *statistical uniqueness* (no adversary can find an alternate solution with non-negligible probability) based on repeated squaring and the soundness of the Fiat-Shamir heuristic for superconstant-round proofs. Later, [EFKP20] constructs a *continuous VDF* satisfying *computational uniqueness* (no computationally bounded adversary can find an alternate solution with non-negligible probability) based on weaker assumptions.

As an effort to close the gap between existing constructions and known lower bounds, we show that:

VDFs with computational uniqueness do not exist in the ROM.

1.1 Our results

In this paper, we provide an *equivalent* reformulation for VDFs in terms of *decision tree algorithms*, which is a different viewpoint towards this question compared to previous works on VDFs [BBBF18; BBF18; Pie19; Wes19; FMPS19; EFKP20; Sta20; MSW20; DGMV20; HHKK23]. This reformulation enables us to use techniques in query complexity to show impossibility results regarding VDFs. Our main result is that if a VDF in the ROM satisfies computational uniqueness, then it cannot satisfy sequentiality.

Theorem 1 (Informal)**.** *Suppose* $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$ *is a VDF in the ROM with computational uniqueness, then 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.*

Theorem 1 implies that VDFs with perfect uniqueness or statistical uniqueness do not exist in the ROM. However, we are able to prove a quantitatively better result regarding VDFs with stronger uniqueness guarantee:

Theorem 2 (Informal)**.** *Suppose* $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$ *is a VDF in the ROM with statistical uniqueness, then there exists an* $O(\mathsf{q}_{\mathsf{Verify}})$*-round* $O(\mathsf{q}^2_{\mathsf{Verify}})$*-query adversary that computes* Eval *correctly with non-negligible probability.*

Notice that the adversary in Theorem 2 that correctly computes Eval only makes $O(\mathsf{q}^2_{\mathsf{Verify}})$ queries, while the adversary in Theorem 1 uses $O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$ queries. We leave as an open question whether one can construct a $O(\mathsf{q}_{\mathsf{Verify}})$-round $\mathsf{poly}(\mathsf{q}_{\mathsf{Verify}})$-query adversary that computes Eval with non-negligible probability when the VDF has computational uniqueness.

We emphasize that Theorem 1 and Theorem 2 only hold in the "pure ROM": security holds against query-bounded adversaries. If VDFs are relaxed to have security against time-bounded adversaries, the adversaries we construct do not suffice anymore: they make few queries to the random oracle but have very large running time. Hence, our results are "tight": [EFKP20] has already constructed a VDF satisfying computational uniqueness in the ROM assuming the hardness of repeated squaring.

1.2 Related works

Impossibility results of VDFs. [MSW20] shows that VDFs with 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 3.5 and Remark 3.6). In fact, we also show a stronger claim regarding VDFs with perfect uniqueness. We postpone a detailed comparison to Section 2.2 and Appendix A. [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. [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.

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 very similar to VDFs because 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 between time-lock puzzles and VDFs is that time-lock puzzles allow the verifier to have knowledge of a *secret key* to achieve efficient verification, while VDFs are publicly verifiable. [MMV11] rules out the possibility to construct 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 only 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 result. 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 start with proving that VDFs with statistical uniqueness in the ROM cannot exist and explain how to generalize this approach to work for computational uniqueness.

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 and deterministically produces an output y. 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) and deterministically decides whether to accept or reject. Verify makes at most $\mathsf{q}_{\mathsf{Verify}}$ queries to f.

The 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 Section 5). The 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 the VDF has *perfect uniqueness* if for every input x, Verify only accepts the output $y := \mathsf{Eval}^f(x)$; the VDF has *statistical uniqueness* if for every input x, Verify accepts an alternative output $y \neq \mathsf{Eval}^f(x)$ with negligible probability; the 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.

Intuitively, fix a security parameter $\lambda \in \mathbb{N}$, for every $x \in \mathcal{X}$, we can define a search problem $S_x \subseteq \{f\colon \{0,1\}^* \to \{0,1\}^\lambda\} \times \mathcal{Y}$ such that $(f, y) \in S_x$ if $\mathsf{Verify}^f(x, y) = 1$. Moreover, we observe that it is sufficient to define S_x as a subset of $[2^\lambda]^n \times \mathcal{Y}$ 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, $\mathbb{D}$ 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 $n \leq |\mathcal{X}| \left(2^{\lambda \mathsf{q}_{\mathsf{Verify}}+1}|\mathcal{Y}| + 2^{\lambda \mathsf{q}_{\mathsf{Eval}}+1}\right)$.

For every $\mathsf{VDF} = (\mathsf{Eval}, \mathsf{Verify})$ and $x \in \mathcal{X}$, we define a search problem $S_x \subseteq [2^\lambda]^n \times \mathcal{Y}$ by a family of verifiers $\{\mathbb{V}_y^{(x)} \colon [2^\lambda]^n \to \{0,1\}\}_{y \in \mathcal{Y}}$ such that for every $y \in \mathcal{Y}$,

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

Moreover, we can define an algorithm $\mathbb{D}^{(x)} \colon [2^\lambda]^n \to \mathcal{Y}$ that computes S_x:

$$\text{For every } f \in [2^\lambda]^n, \mathbb{D}^{(x)}(f) = \mathsf{Eval}^f(x).$$

Observe that for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $\mathbb{V}_y^{(x)}$ has query complexity $\mathsf{q}_{\mathsf{Verify}}$ and $\mathbb{D}^{(x)}$ has query complexity $\mathsf{q}_{\mathsf{Eval}}$. 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.

Lemma 1 (VDFs with perfect uniqueness don't exist in the ROM)**.** *Suppose* $\mathsf{VDF} = (\mathsf{Verify}, \mathsf{Eval})$ *is a VDF in the ROM with perfect completeness and perfect uniqueness, then there exists an* $O(\mathsf{q}_{\mathsf{Verify}})$*-round* $O(\mathsf{q}_{\mathsf{Verify}}^2)$*-query adversary that computes* Eval *correctly with probability* 1.

To prove Lemma 1, we consider the search problem reformulation outlined in Section 2.1. Hence, to show that VDF does not satisfy sequentiality, we show that for every search problem S_x for $x \in \mathcal{X}$, there is a $\mathsf{q}_{\mathsf{Verify}}$-round $\mathsf{q}_{\mathsf{Verify}}^2$-query adversary $\mathbb{A}^{(x)}$ that solves S_x.

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 $y \in \mathcal{Y}$ such that $(f, y) \in S_x$. Hence, the solution y can be fully determined by the verifiers. In fact, we can construct our adversary solely from the verification algorithms.

Intuitively speaking, for a fixed input $x \in \mathcal{X}$, let's consider the set of all accepting leaves $\{\ell_i\}_i$ of the verifiers $\{\mathbb{V}_y^{(x)}\}_{y \in \mathcal{Y}}$. 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 $y \in \mathcal{Y}$. 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:

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

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

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

Since the VDF has perfect uniqueness, we know that every oracle function $\ell \in [2^\lambda]^n$ has a unique solution, which implies that $\mathsf{Cube}(\ell_i)$'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 algorithm that always outputs the correct solution y and makes at most $\mathsf{q}_{\mathsf{Verify}}^2$ queries in $\mathsf{q}_{\mathsf{Verify}}$ rounds.

The algorithm described above can be summarized as follows:

1. Let L be the set of all accepting leaves $\{\ell_i\}_i$ of the verifiers $\{\mathbb{V}_y^{(x)}\}_{y\in\mathcal{Y}}$.
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 $\ell \in \mathsf{Cube}(p^*)$.

We leave the detailed analysis to Appendix A.

Remark 1. Note that Lemma 1 is proven in [MSW20] by constructing an adversary that computes Eval with $2\mathsf{q}_{\mathsf{Verify}} + 1$ rounds and $(2\mathsf{q}_{\mathsf{Verify}} + 1) \cdot \mathsf{q}_{\mathsf{Eval}}$ queries. Qualitatively they show a similar result as in our Lemma 1: VDFs with perfect completeness and perfect uniqueness cannot exist in the ROM. However, we construct a sequentiality adversary using only $\mathsf{q}_{\mathsf{Verify}}$ rounds and $\mathsf{q}_{\mathsf{Verify}}^2$ queries. Moreover, our construction still works when VDFs have imperfect completeness (see Appendix A).

Remark 2. The proof of Lemma 1 is in essence the classical algorithm witnessing that *decision tree complexity* is at most the square of *certificate complexity* for total boolean functions ([AB09]).

The breakthrough result of Kahn, Saks, and Smith [KSS11] enables a similar algorithm to work in the randomized setting. In Section 4 using this algorithm we prove Theorem 2.

2.3 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.

2.3.1 Adversary for VDFs with statistical uniqueness in the ROM

In this section we present another proof of Theorem 2 with the additional assumption that the completeness is perfect. This proof yields a quantitatively worse algorithm, but we show that this algorithm can be modified to work for computationally unique VDFs.

Similar to Section 2.1 we employ our search problems language for VDFs. Note that now the VDF only satisfies statistical uniqueness, we don't expect our sequentiality adversary to perfectly compute Eval anymore. Rather, to show that VDF does not satisfy sequentiality, we show that there exists some constant C such that for every $x \in \mathcal{X}$, there is a $O(\mathsf{q}_{\mathsf{Verify}})$-round $O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$-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 the 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 $2\mathsf{q}_{\mathsf{Verify}} + 1$ rounds to compute $\mathbb{D}^{(x)}$. 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 $\mathbb{D}^{(x)}(f)$, it queries at least one "new" position that has also been queried by $\mathbb{V}^{(x)}_{\mathbb{D}^{(x)}(f)}(f)$ in this iteration. Formally, let $y = \mathbb{D}^{(x)}(f)$ and $\ell_{\mathbb{V},f}$ be the unique accepting leaf of $\mathbb{V}_y^{(x)}$ that contains f. Since VDF satisfies perfect uniqueness, which means

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

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

1: Let $L_1 \coloneqq \{\ell_i\}_i$ be the set of leaves of $\mathbb{D}^{(x)}$.
2: Initialize $p^* \coloneqq \star^n$.
3: Initialize $W \coloneqq [\,]$.
4: **for** $i \in [2\mathsf{q}_{\mathsf{Verify}} + 1]$ **do**
5: Choose an arbitrary leaf ℓ_i from L_i.
6: Append $y_i \coloneqq \mathbb{D}^{(x)}(\ell_i)$ to W.
7: For every $q \in \mathsf{dom}(\ell_i)$, query f at q and set $p^*[q] \coloneqq f[q]$.
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.

that for every chosen leaf ℓ_i such that $y_i = \mathbb{D}^{(x)}(\ell_i) \neq \mathbb{D}^{(x)}(f)$, let p_i^* be the value of p^* at the beginning of iteration i, 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 $\mathbb{V}^{(x)}_{\mathbb{D}^{(x)}(f)}$ has query complexity at most $\mathsf{q}_{\mathsf{Verify}}$, at most $\mathsf{q}_{\mathsf{Verify}}$ of the recorded solutions are not equal to $\mathbb{D}^{(x)}(f)$, which implies that the majority of recorded solutions is always $\mathbb{D}^{(x)}(f)$.

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

To be more specific, if the VDF does not have perfect uniqueness, in a round i that $\mathbb{A}^{(x)}(f)$ chooses a leaf ℓ_i that leads to some solution $y' \neq \mathbb{D}^{(x)}(f)$, 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 simple 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 $\mathsf{q}_{\mathsf{Verify}}$ 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 $y \in \mathcal{Y}$ where $y \neq \mathbb{D}^{(x)}$ and $\mathbb{V}^{(x)}_y = 1$. 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.

2.3.2 Does computational uniqueness undermine the adversary?

We briefly discuss how the above adversary $\mathbb{A}^{(x)}$ would still succeed even when the VDF satisfies only computational uniqueness. (We do need to modify $\mathbb{A}^{(x)}$ further for technical reasons, but the version outlined in Section 2.3.1 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 2.3.1 fails after relaxing 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 $y \in \mathcal{Y}$ where $y \neq \mathbb{D}^{(x)}$ and $\mathbb{V}_y^{(x)} = 1$.
- *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 $y \in \mathcal{Y}$ where $y \neq \mathbb{D}^{(x)}$ and $\mathbb{V}_y^{(x)} = 1$.

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 2.3.1 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 2.3.1, 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 the 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 $\mathcal{Y}$.*
- $\mathsf{Eval}^f(\mathsf{pp}, x) \to y$*: On input the public parameter* pp *and an element $x \in \mathcal{X}$, the* **deterministic** *evaluation algorithm* Eval *outputs y.*
- $\mathsf{Verify}^f(\mathsf{pp}, x, y) \to \{0,1\}$*: On input the public parameter* pp*, and element $x \in \mathcal{X}$, and a value $y \in \mathcal{Y}$, 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 $\mathsf{q}_{\mathsf{Verify}} \ll \mathsf{q}_{\mathsf{Eval}}$.*

Remark 3.3. In the VDF definition given in [BBBF18], Eval is an algorithm that outputs y and a proof π, in which y is generated deterministically but π can be generated in a randomized manner. We omit the output proof π in Definition 3.2 for simplicity. However, all our results can be straightforwardly extended to the setting where Eval has an additional output π by fixing the internal randomness of Eval in a way such that the completeness error is minimized.

Definition 3.4 (Completeness of VDF)**.** $\mathsf{VDF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ *has completeness error α if for every $\lambda \in \mathbb{N}$ and $\mathsf{q}_{\mathsf{Eval}} \in \mathbb{N}$,*

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

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

Definition 3.5 (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*,*

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

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 3.6. 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 3.7 ($(\mathsf{r}_{\mathsf{Adv}}, \mathsf{q}_{\mathsf{Adv}}, \gamma)$-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,*

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

Remark 3.8. 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 3.9. *A* **search problem** *is defined by a relation $S \subseteq \mathrm{F} \times \mathrm{Y}$, or equivalently, a family of verifiers $\{\mathbb{V}_y\}_{y \in \mathrm{Y}}$ such that $(f, y) \in S$ if and only if $\mathbb{V}_y(f) = 1$. We say a search problem is* **total** *if, for every $f \in \mathrm{F}$, there is at least one solution y s.t. $(f, y) \in S$.*

Definition 3.10. *Given a search problem (not necessarily total) S defined by $S \subseteq \mathrm{F} \times \mathrm{Y}$. We say an adversary $\mathbb{D} \colon \mathrm{F} \to \mathrm{Y}$ $(1-\alpha)$-***approximately computes*** S, if*

$$\Pr_{f \leftarrow cF}[(f, \mathbb{D}(f)) \in S] \geq 1 - \alpha.$$

We focus on search problems with input space $\mathrm{F} = [M]^m$ for $M, m \in \mathbb{N}$.

Definition 3.11 (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 $\mathsf{dom}(p) \coloneqq \{i : p_i \neq \star\}$.

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 $\mathsf{Cube}(p) \coloneqq \{f \in [M]^m : \forall\, i \in \mathsf{dom}(p), f[i] = p[i]\}$ the set of all inputs consistent with p. Note that $\mathsf{Cube}(p)$ is a cube.

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 $\mathsf{Cube}(p) \cap \mathsf{Cube}(q) = \mathsf{Cube}(p \cup q)$.

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 $p_i \coloneqq \star$.

Let $g\colon \mathrm{F} \to \mathrm{Y}, g'\colon \mathrm{F}' \to \mathrm{Y}$ be two functions. We say g' is a ***subfunction*** of g, if $\mathrm{F}' \subseteq \mathrm{F}$ and $g(f) = g'(f)$ for all $f \in \mathrm{F}'$.

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 $S_{\ell,x} \subseteq \mathsf{Cube}(\ell) \times \mathcal{Y}$ as follows:

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

ii. There is an algorithm $\mathbb{D}\colon \mathsf{Cube}(\ell) \to \mathcal{Y}$ of query complexity $\mathsf{q}_{\mathsf{Eval}}$ that $(1-\alpha_{\ell,x})$-approximately computing $S_{\ell,x}$ which satisfies that $\mathbb{D}(f) = \mathsf{Eval}^f(\mathsf{pp}, x)$ 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 3.4 that

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

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 *satisfying statistical uniqueness with error* ϵ *such that* $\alpha + \epsilon < 0.1$*. 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* VDF *does not satisfy* $(\mathsf{q}_{\mathsf{Setup}} + O(\mathsf{q}_{\mathsf{Verify}}), \mathsf{q}_{\mathsf{Setup}} + O(\mathsf{q}_{\mathsf{Verify}}^2), 1 - 2\sqrt{\alpha + \epsilon})$*-sequentiality.*

According to Section 3.3, it suffices to prove the following lemma.

Lemma 4.2. *Let* $S \subseteq [M]^m \times \mathrm{Y}$ *be a search problem such that there exist a family of verifiers* $\{\mathbb{V}_y\}_{y \in \mathrm{Y}}$ *of query complexity* t *where*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\textit{there exists unique } y \in \mathrm{Y} \textit{ such that } \mathbb{V}_y(\boldsymbol{f}) = 1] \geq 1 - \epsilon - \alpha > 0.9.$$

Then there exists an $O(t)$*-round and* $O(t^2)$*-query algorithm* $\mathbb{A} \colon [M]^m \to \mathrm{Y}$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{V}_{\mathbb{A}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq 1 - 2\sqrt{\epsilon + \alpha}.$$

Proof of Theorem 4.1 by Lemma 4.2. Let $\mathbb{A}$ be the adversary in Lemma 4.2. We can 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 $S_{\ell_f, x}$ as defined in Section 3.3. If follows that Adv is a $(\mathsf{q}_{\mathsf{Setup}} + O(\mathsf{q}_{\mathsf{Verify}}))$-round $(\mathsf{q}_{\mathsf{Setup}} + O(\mathsf{q}_{\mathsf{Verify}}^2))$-query algorithm that correctly computes Eval with probability at least $1 - 2\sqrt{\epsilon + \alpha}$ as desired. □

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

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

Let L be the collection of all the accepting leaf subcubes of $\mathbb{V}_y$ across all $y \in \mathrm{Y}$.
Initialize $p^* \coloneqq \star^m$.
$i \coloneqq 0$ ▷ We bound the number of iterations, to have the algorithm always terminate.
while $\forall \ell \in L \colon \mathsf{Cube}(p^*) \not\subseteq \mathsf{Cube}(\ell)$ **do**
  if $\Pr_{g \leftarrow \mathsf{Cube}(p^*)}[g \in U(L)] < 1 - \sqrt{\delta}$ **then**
    Output $\perp$.
  $i \coloneqq i + 1$;
  if $i > t$ **then break** ▷ t to be chosen later.
  Choose the cube $\ell \in L$ according to Theorem 4.3.
  For every $q \in \mathsf{dom}(\ell)$, query f at q and set $p^*[q] \coloneqq f[q]$.
  Remove from L all the cubes disjoint with $\mathsf{Cube}(p^*)$.
return the value y corresponding to the subcube in L containing $\mathsf{Cube}(p^*)$.

Our key tool is the following theorem.

Theorem 4.3 ([KSS11])**.** *Let* $C_1, \ldots, C_k$ *be a collection of subcubes of* $[M]^m$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\textit{there exists unique } i \in [k] \textit{ such that } \boldsymbol{f} \in C_i] \geq 1 - \delta.$$

Then there exists $i_0 \in [k]$ *such that*

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\boldsymbol{f} \in C_{i_0} \ \vee \ \textit{there exists unique } j \in [k] \textit{ such that } \boldsymbol{f} \in C_j \textit{ and } C_{i_0} \cap C_j = \emptyset] \geq 1 - 10\delta.$$

The algorithm and the proof below essentially repeat [Rud88, Theorem 9.2].

Proof of Lemma 4.2. For a collection of subcubes L define $U(L)$ to be the subset of $[M]^m$ uniquely covered by L. Let $\delta := \epsilon + \alpha$. The adversary $\mathbb{A}$ is described in Algorithm 2.

First observe that Line 6 is executed with probability at most $\sqrt{\delta}$. As otherwise the complement of $U(L)$ covers more than δ fraction of $[M]^m$.

Let $\ell^* \in L$ be the subcube from L containing f. We measure the progress of our algorithm by the number of coordinates in $\mathsf{dom}(\ell^*)$ we query, as soon as we query all of them, the condition in Line 4 is violated and the algorithm successfully terminates. By construction $|\mathsf{dom}(\ell^*)| \leq t$. At each iteration we query the (not queried previously) elements of $\mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)$. So we make progress in the iteration whenever $(\mathsf{dom}(\ell) \setminus \mathsf{dom}(p^*)) \cap \mathsf{dom}(\ell^*) \neq \emptyset$. By Theorem 4.3 we have that this happens with probability $1 - 10\delta$. Let $\mathbf{k}_i$ be a random variable that equals 1 if at the i-th iteration $(\mathsf{dom}(\ell) \cap \mathsf{dom}(\ell^*)) \setminus \mathsf{dom}(p^*) \neq \emptyset$ and 0 otherwise. Then $\mathbb{E}[\mathbf{k}_i] \geq 1 - 10\delta$ by Theorem 4.3. Then $\mathbb{E}\left[\sum_{i \in [t]} \mathbf{k}_i\right] \geq (1 - 10\delta)t$. Hence by Markov's inequality for $\sum_{i \in [t]} \mathbf{k}_i$ with $t = \mathsf{q}_{\mathsf{Verify}}/((1 - 10\delta) \cdot (1 - \sqrt{\delta}))$ we get that after t iterations we query all values of f in $\mathsf{dom}(\ell^*)$ with probability $1 - \sqrt{\delta}$. Hence, another $\sqrt{\delta}$ error is introduced in Line 8. The total error is then $2\sqrt{\delta}$ as required. □

5 VDFs with computational uniqueness

We discuss VDFs with computational uniqueness in the ROM.

Theorem 5.1 (Lower bounds for VDFs with computational uniqueness)**.** *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* $\mathsf{q}_{\mathsf{Adv}} = O(\mathsf{q}_{\mathsf{Verify}} \cdot \mathsf{q}_{\mathsf{Eval}})$.

Analogously to Section 4, it suffices to prove the following lemma about search problems to derive Theorem 5.1,

Lemma 5.2. *Let* $S \subseteq [M]^m \times \mathrm{Y}$ *be a search problem, determined by verifiers* $\mathbb{V}$ *of query complexity at most* t*. Let* $\mathbb{D} \colon [M]^m \to \mathrm{Y}$ *be an algorithm of query complexity* T *that* $(1-\alpha)$*-approximately computes* S*. 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*

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

2. *there exists an adversary* $\mathbb{B}$ *of query complexity* $O(t'T)$ *such that*

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

Proof of Theorem 5.1 by Lemma 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 Lemma 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

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

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

$$\Pr\left[\begin{array}{l} y \neq \mathsf{Eval}^f(\mathsf{pp}, x) \\ \wedge\, \mathsf{Verify}^f(\mathsf{pp}, x, y) = 1 \end{array} \;\middle|\; \begin{array}{r} f \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \mathsf{q}_{\mathsf{Eval}}) \\ x \leftarrow \mathcal{X} \\ y \leftarrow \mathsf{Adv}^f(\mathsf{pp}, x) \end{array} \right] \geq \epsilon. \tag{3}$$

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

□

5.1 The sequentiality breaker

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

Algorithm 3 Adversary $\mathbb{A}$, the sequentiality breaker.

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

$p^* \coloneqq \star^m$.
$K \coloneqq \emptyset$.
for $r \in [t']$ **do**
    Uniformly sample $f^* \in [M]^m$ consistent with p^*.
    Let ℓ be the unique leaf of $\mathbb{D}$ such that $\mathsf{Cube}(\ell)$ contains f^*.
    $K \coloneqq K \uplus \{\text{the solution associated with } \ell\}$.
    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.
return the majority of solutions in K if it exists; $\perp$ otherwise.

It is not hard to see $\mathbb{A}$ 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 additional notation.

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

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 $\mathbb{A}(f) = \mathbb{D}(f)$ with high probability by simple Markov's inequality. Now we 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* $\epsilon \coloneqq \Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f}} = 1]$. *Then* $\Pr_{\boldsymbol{f}}[\mathbb{A}(\boldsymbol{f}) \neq \mathbb{D}(\boldsymbol{f})] \leq \frac{2t'}{t'-2t}\epsilon + \alpha$.

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 $\sum_{r=1}^{t'} \boldsymbol{S}^{(1)}_{r,f}$ with the assumption that $\Pr_{\boldsymbol{f},\boldsymbol{r}}[\boldsymbol{f} \in \overline{\mathrm{F}} \wedge \boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f}}]$ is small. Let ϵ' denote $\frac{2t'}{t'-2t}\epsilon$. 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 $\alpha = 1 - |\overline{\mathrm{F}}|/M^m$), we have $f \in \overline{\mathrm{F}}$ and $\sum_{r=1}^{t'} \boldsymbol{S}^{(2)}_{r,f} < t\epsilon/\epsilon' = t'/2 - t$. For those f, $\sum_{r=1}^{t'} \boldsymbol{S}^{(1)}_{r,f} + \boldsymbol{S}^{(2)}_{r,f} < t'/2$. Thus the majority of recorded solutions are exactly $\mathbb{D}(f)$. We conclude that the algorithm succeeds with probability at least $1 - \epsilon' - \alpha$. □

5.2 The uniqueness breaker

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

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

Proof. We construct the adversary $\mathbb{B}$ in Algorithm 4. Through the execution of $\mathbb{B}$, we can define $\mathcal{D}_{\mathbb{B}}$

Algorithm 4 Adversary $\mathbb{B}$, the uniqueness breaker.

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

1: Run $\mathbb{D}(f)$, let I be the set of indices queried during the execution.
2: $J \coloneqq \emptyset$.
3: $p^* \coloneqq \star^m$.
4: $K \coloneqq \emptyset$.
5: **for** $r \in [t']$ **do**
6: Uniformly sample $p' \leftarrow [M]^{I \setminus \mathsf{dom}(p^*)}$.
7: $f' \coloneqq f_{(I \setminus \mathsf{dom}(p^*)) \to p'} \cdot$
8: $y' \coloneqq \mathbb{D}(f')$.
9: **if** $y' \neq \mathbb{D}(f) \wedge \mathbb{V}_{y'}(f') = 1$ **then return** y'.
10: Uniformly sample $f^* \in [M]^m$ consistent with p^*.
11: Let ℓ be the unique leaf of $\mathbb{D}$ such that $\mathsf{Cube}(\ell)$ contains f^*.
12: 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.
13: **return** $\perp$

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 7). See Fig. 1 for visualization.

To prove the lemma, it suffices to show

$$\Pr_{(\boldsymbol{r},\boldsymbol{p^*},\boldsymbol{f},\boldsymbol{f'}) \leftarrow \mathcal{D}_{\mathbb{B}}}[\mathbb{D}(\boldsymbol{f'}) \neq \mathbb{D}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f'})}(\boldsymbol{f})] \geq \epsilon. \tag{4}$$

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 $J \coloneqq \mathsf{dom}(\ell) \setminus \mathsf{dom}(\boldsymbol{p^*})$ denote the set of indices fixed by ℓ but not by $\boldsymbol{p^*}$ and we can define $\boldsymbol{f} \coloneqq f_{J \to \ell_J}$.

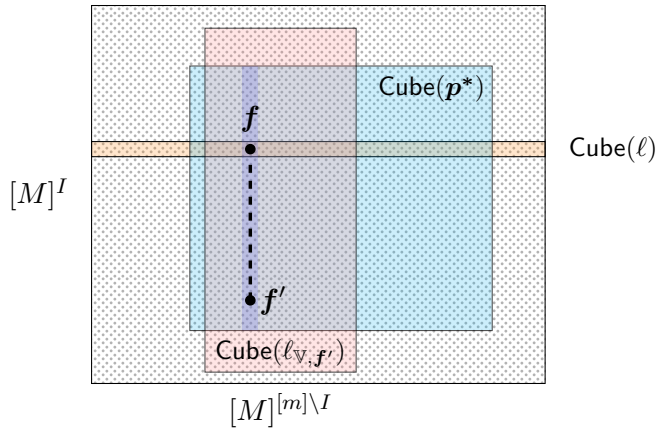

Figure 1: Distribution $\mathcal{D}_{\mathbb{B}}$.

Now observe that if $\boldsymbol{f}' \in \overline{\mathrm{F}}$ and $\boldsymbol{S}^{(2)}_{\boldsymbol{r},\boldsymbol{f}'} = 1$, then $\mathbb{D}(\boldsymbol{f}') \neq y$ and $\mathsf{Cube}(\boldsymbol{p}^* \cup \ell) \cap \mathsf{Cube}(\boldsymbol{p}^* \cup \ell_{\mathbb{V},\boldsymbol{f}'}) \neq \emptyset$. 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 $\mathbb{D}(\boldsymbol{f}') \neq \mathbb{D}(\boldsymbol{f})$ 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 $\mathbb{D}(\boldsymbol{f}') \neq \mathbb{D}(\boldsymbol{f}) \wedge \mathbb{V}_{\mathbb{D}(\boldsymbol{f}')}(\boldsymbol{f}) = 1$, we can deduce that

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

Thus to prove (4), it suffices to show $\mathcal{D}_{\mathbb{B}} \equiv \mathcal{D}_{\mathbb{A}}$, that is, for every $r \in [t'], p^* \in ([M] \cup \{\star\})^m, f, f' \in [M]^m$,

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

To this end, we need the following four statements.

Claim 5.5. *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 5.6. *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 5.7. *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 5.8. *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 $f \in \mathsf{Cube}(p^*)$, $\Pr_{\mathcal{D}_{\mathbb{A}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f] = \Pr_{\mathcal{D}_{\mathbb{B}}}[\boldsymbol{f'} = f' \mid \boldsymbol{r} = r', \boldsymbol{p^*} = p^*, \boldsymbol{f} = f]$.*

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 chain rule, we deduce that $\mathcal{D}_{\mathbb{A}} \equiv \mathcal{D}_{\mathbb{B}}$. □

5.3 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 5.9. *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 5.9 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 5.9, it suffices to prove the following lemma:

Lemma 5.10. *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 $y \in \mathrm{Y}$ such that $(f, y) \in S$ but $y \neq \mathbb{D}(f)$.
(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 5.10, we define the following hard (on average) functions against parallel decision trees.

Definition 5.11. *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.*

Claim 5.12. *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 $Q^\star \leq M^l$ 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 $R = k - 1 \geq 0$. 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 $1 \leq v \leq M^\ell$, let I_v be the set of indices of variables that appear in the subtree with root w_v, g_v be the subfunction of $h_{M,k\ell+1}$ with domain $\mathsf{Cube}(w_v)$. Let $V \coloneqq \{v : I_v \cap I_0 \neq \emptyset\}$. By the definition of $h_{M,k\ell+1}$, $I_1, \ldots, I_{M^\ell}$ are pairwise disjoint, so $|V| \leq |I_0| = Q_0$.

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 $h_{M,k\ell+1}$:

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

$$
\begin{aligned}
&\leq \frac{1}{M^\ell}\left(|V| + (M^\ell - |V|)(1 + (Q^\star - Q_0)/M^\ell)/2\right) \\
&\leq \frac{1}{M^\ell}\left(Q_0 + (M^\ell - Q_0)(1 + (Q^\star - Q_0)/M^\ell)/2\right) \\
&\leq (1 + Q^\star/M^\ell)/2.
\end{aligned}
$$

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

Proof of Lemma 5.10. 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 $\{f : f_1 \neq 1\}$, $\mathbb{D}$ 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

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

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 Claim 5.12, (iii) holds. □

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.

A VDFs with perfect uniqueness

Lemma A.1. *Suppose* $\mathsf{VDF} = (\mathsf{Verify}, \mathsf{Eval})$ *is a VDF in the ROM with completeness error* α *satisfying perfect uniqueness. 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* VDF *does not satisfy* $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}, \mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}^2, 1-\alpha)$*-sequentiality.*

Similar to Section 4, it suffices to prove the following lemma.

Lemma A.2. *Let* $S \subseteq [M]^m \times \mathrm{Y}$ *be a search problem such that there exists a family of verifier* $\{\mathbb{V}_y\}_{y\in\mathrm{Y}}$ *of query complexity* t *where*

$$\Pr_{\boldsymbol{f}\leftarrow[M]^m}[\textit{there exists unique } y \in \mathrm{Y} \textit{ such that } \mathbb{V}_y(\boldsymbol{f}) = 1] \geq 1-\alpha.$$

Then there exists an t*-round and* t^2*-query adversary* $\mathbb{A}\colon [M]^m \to \mathrm{Y}$ *such that*

$$\Pr_{\boldsymbol{f}\leftarrow[M]^m}[\mathbb{V}_{\mathbb{A}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq 1-\alpha.$$

Proof of Lemma A.1 by Lemma A.2. Let $\mathbb{A}$ be the adversary in Lemma A.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 $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}})$-round $(\mathsf{q}_{\mathsf{Setup}} + \mathsf{q}_{\mathsf{Verify}}^2)$-query algorithm that correctly computes Eval with probability at least $1-\alpha$. □

Proof of Lemma A.2. We construct $\mathbb{A}$ as follows.

Algorithm 5 Sequentiality-breaking adversary $\mathbb{A}$ for perfect uniqueness.

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

1: Initialize $L_1 \coloneqq \{\ell_1, \ldots, \ell_k\}$ to be the set of all partial assignments corresponding to the accepting leaves of all verifiers $(\mathbb{V}_y)_{y\in\mathrm{Y}}$.
2: Initialize $p^* \coloneqq \star^n$.
3: **for** $i \in [t]$ **do**
4: **if** $L_i = \emptyset$ **then**
5: Output $\perp$.
6: Choose an arbitrary partial assignment ℓ_i from L_i.
7: For every $q \in \mathsf{dom}(\ell_i)$, query f at q and set $p^*[q] \coloneqq f[q]$.
8: Let $L_{i+1} \subseteq L_i$ be the set of all leaves in L_i that are consistent with p^*.
9: **return** $y \in \mathrm{Y}$ such that for all $\ell \in \mathsf{Cube}(p^*)$, $\mathbb{V}_y(\ell) = 1$ if it exists; $\perp$ otherwise.

We argue that (i) $\mathbb{A}$ is a t-round t^2-query algorithm, and (ii) $\mathbb{A}$ outputs y such that $y \neq \perp$ and $\mathbb{V}_y(f) = 1$ with probability $1-\alpha$.

Running time of $\mathbb{A}$. $\mathbb{A}$ makes t rounds of queries to f. Note that because the verifiers $\{\mathbb{V}_y\}_y$ has query complexity at most t, the number of positions $q \in [n]$ such that $\ell[q] \neq \star$ is at most t for each $\ell \in L_1$. Therefore, in each round, $\mathbb{A}$ makes at most t queries to f.

Correctness of $\mathbb{A}$. Let $f \in [M]^m$ be an input such that there exists a unique $y \in \mathrm{Y}$ such that $\mathbb{V}_y(f) = 1$.

For every $i \in [t]$, let p_i^* be the partial assignment p^* at the beginning of iteration i. Note that p_i^* is consistent with every partial assignment $\ell \in L_i$. We define c_i to be the maximum number of entries fixed by ℓ but not by p_i^* over all $\ell \in L_i$; in other words,

$$c_i \coloneqq \max_{\ell \in L_i} \left|\{k \in \mathsf{dom}(\ell) : p_i^*[k] = \star\}\right|.$$

Notice that $c_1 \leq t$ because $p_1^* = \star^m$ and $\mathsf{dom}(\ell) \leq t$ for every $\ell \in L_1$.

We claim that for every $i \in [t]$, $c_{i+1} \leq \max\{0, c_i - 1\}$.

Let ℓ_i^* be the partial assignment chosen during iteration i. Since every f has a unique solution, we know that $\{\mathsf{Cube}(\ell)\}_{\ell \in L_i}$ form a partition of $\mathsf{Cube}(p_i^*)$. Hence, for every $\ell_u \neq \ell_v \in L_i$, there exists some $k \in \mathsf{dom}(\ell_u) \cap \mathsf{dom}(\ell_v)$ such that $\ell_u[k] \neq \ell_v[k]$, as otherwise we can infer $\mathsf{Cube}(\ell_u) \cap \mathsf{Cube}(\ell_v) = \emptyset$.

Let ℓ be an arbitrary partial assignment in $L_{i+1} \subseteq L_i$, we consider the following two cases:

- $\ell \neq \ell_i^*$. We know there exists some $k \in \mathsf{dom}(\ell) \cap \mathsf{dom}(\ell_i^*)$ such that $\ell[k] \neq \ell_i^*[k]$. Moreover, $p_i^*[k] = *$, as otherwise, one of ℓ, ℓ_i^* is inconsistent with p_i^*. Note that since $k \in \mathsf{dom}(\ell_i^*)$, $f[k]$ is queried in the i-th iteration. We can then deduce that

$$\left|\{k \in \mathsf{dom}(\ell) : p_{i+1}^*[k] = \star\}\right| \leq \left|\{k \in \mathsf{dom}(\ell) : p_i^*[k] = \star\}\right| - 1.$$

- $\ell = \ell_i^*$. We have $\left|\{k \in \mathsf{dom}(\ell_i^*) : p_{i+1}^*[k] = \star\}\right| = 0$ since all $q \in \mathsf{dom}(\ell_i^*)$ is queried in the i-th iteration. We can thus conclude that $c_{i+1} \leq \max\{0, c_i - 1\}$.

Therefore, after the i-th iteration, every partial assignment in L_{i+1} has at most $t - i$ entries not fixed by p^*. Namely, for every $\ell \in L_{i+1}$,

$$\left|\{k \in \mathsf{dom}(\ell) : p^*[k] = \star\}\right| \leq t - i.$$

At the end of the t-th round, all remaining partial assignments ℓ in L_t satisfies the following:

$$\forall\, k \in \mathsf{dom}(\ell), p^*[k] = \ell[k],$$

then there must be a common solution y for all $\ell \in \mathsf{Cube}(p^*)$ that $\mathbb{A}(f)$ outputs.

Hence, for every $f \in [M]^m$ where there exists unique $y \in \mathrm{Y}$ such that $\mathbb{V}_y(f) = 1$, $\mathbb{A}$ correctly compute y. In other words,

$$\Pr_{\boldsymbol{f} \leftarrow [M]^m}[\mathbb{V}_{\mathbb{A}(\boldsymbol{f})}(\boldsymbol{f}) = 1] \geq 1 - \alpha. \qquad \square$$

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.

[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 Nalini 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.

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

[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.