

It Takes Two: Proofs of Work for Fiat–Shamir

Benedikt Bünz

bb@nyu.edu

NYU

Jessica Chen

jessicachen@nyu.edu

NYU

Ziyi Guan

ziyiguan@mit.edu

MIT & NYU

October 3, 2026

Abstract

In the random oracle model (ROM), proof of work (PoW) serves two roles in the Fiat–Shamir transformation. First, in challenge grinding, the prover must solve a puzzle for each candidate challenge, which amplifies soundness and reduces proof size and verifier time. This requires a *non-amortizing* PoW: computing k distinct accepted puzzle–solution–proof triples costs, in expectation, roughly k times the work of computing one. Second, PoW can prevent diagonalization attacks, in which the statement’s circuit computes its own challenge. The XFS transformation of Arnon and Yagev (CRYPTO 2025) uses a *strong* PoW: an adversary that does substantially less work than the honest prover solves a random puzzle with only negligible probability, even after bounded preprocessing.

Can a single PoW be both strong and non-amortizing? We prove that it cannot, whenever the verifier is sufficiently cheaper than the prover. Our main technical tool converts computational uniqueness into statistical uniqueness without increasing verifier query complexity. Combined with the search bound of Guan, Riazanov, and Yuan (CRYPTO 2025), which builds on Smyth (STOC 2002), it resolves their open question.

To overcome this barrier, our XFS-with-grinding transformation for relativized Σ -protocols composes a strong PoW with a non-amortizing one. The resulting non-interactive argument retains the soundness amplification of grinding and the security guarantee of XFS in the relativized ROM, with additive prover work for the two PoWs.

Keywords: Fiat–Shamir transformation, proof of work, challenge grinding, diagonalization, random oracle model

Contents

1	Introduction	3
1.1	Our results	4
1.2	Related work	5
2	Technical overview	7
2.1	Impossibility of a strong and non-amortizing PoW	7
2.2	XFS with grinding	11
3	Preliminaries	12
3.1	Relativized Σ -protocol	12
3.2	Prefix-avoiding padder	13
3.3	Proof of work	14
4	From non-amortization to computational uniqueness	18
5	From computational uniqueness to statistical uniqueness	22
5.1	Technical lemmas	23
5.2	Proof of Theorem 5.3	27
6	Impossibility of PoW	30
6.1	Chaining the two reductions together	30
6.2	Proof of Theorem 6.1	32
7	XFS with grinding	34
7.1	Proof of Theorem 7.2	35
7.2	Proof of Lemma 7.4	37
7.3	Proof of Lemma 7.5	39
7.4	Proof of Lemma 7.6	41
7.5	Proof of Lemma 7.7	41
7.6	Proof of Lemma 7.8	42
	Acknowledgments	45
	AI disclosure	45
	References	45

1 Introduction

The Fiat–Shamir (FS) transformation [FS87] is a generic way to remove interaction from interactive arguments in the random oracle model (ROM) [BR93]. Consider a Σ -protocol for a statement x : the prover P sends the first message m_1 , the verifier V replies with a random challenge ρ , and P answers with m_2 . FS removes the verifier’s message by setting ρ to the random oracle’s output on (x, m_1) , so the prover computes the challenge itself. In this work, we study two applications of proof of work (PoW) [DN93] to FS. In a PoW scheme, the prover P receives a puzzle and spends computational work to solve it, outputting a solution and possibly an auxiliary proof, and the verifier V can check a solution far more cheaply than it can be found. The work required to solve a puzzle, measured in random oracle queries, is the PoW’s *difficulty*, which we denote by D .

Soundness amplification. Suppose the interactive protocol has soundness error κ . In the ROM, applying FS generally loses a factor of q , where q is the number of random oracle queries made by the adversary. The adversary can try many first messages and choose one that yields a favorable challenge. A natural way to make this harder for the adversary is to require it to solve a puzzle for each candidate challenge. This technique is known as *challenge grinding*. If each new candidate challenge requires D fresh queries in expectation, then grinding reduces the leading soundness term from $q \cdot \kappa$ to roughly $(q/D) \cdot \kappa$. An adversary obtains a new candidate challenge either by changing the puzzle or by finding a different solution to the same puzzle, both of which require fresh queries, while the honest prover makes only D additional queries in expectation [Sta21; BT24]. Since the increase in prover cost is only additive, grinding with a difficulty D well below the prover’s own cost is cheaper than amplifying soundness by repetition, which increases the prover cost multiplicatively. This grinding approach is widely deployed in hash-based proof systems, where amplification of low-soundness *query* rounds is generally required [BBHR18; ACFY24; ACFY25; ZCF24; BFRW25]. In fact, essentially every hash-based SNARK library [Sta21; Pol22; DP26; BRW26] uses grinding. In these deployments, grinding reduces the proof size and verifier time by 15–25% at a fixed security level [Sta21; Pol22].

Preventing diagonalization attacks. A different concern with FS is that security in the ROM need not hold after instantiating the random oracle with a concrete hash function [CGH98; GK03]. Khovratovich, Rothblum, and Soukhanov [KRS25] show that applying FS to a practical succinct argument based on the GKR protocol [GKR08] is insecure with any concrete instantiation of the hash function. Their *diagonalization attack* computes the FS challenge inside the circuit being proven and then produces a response tailored to that challenge. The attack is possible because the circuit being proven can evaluate the hash function used for Fiat–Shamir; such a protocol is called *relativized*, since the statement is about a computation that uses the same hash function that generates the challenge. While relativized arguments cannot be proven secure in the pure random oracle model [BCG25], they are widely deployed in practice, especially in the context of recursive proofs [Val08; CT10; BCCT13; BCTV14]. To study the diagonalization attack and provide defenses, Arnon and Yogev [AY25] consider a *relativized random oracle*. They introduce the extended Fiat–Shamir transformation (XFS), which uses PoW to make challenge generation more expensive than evaluating the circuit, so that the circuit cannot compute the challenge. XFS is proven secure in the relativized ROM, and it increases the prover’s work additively by one PoW whose difficulty exceeds the cost q_C of evaluating the circuit.

One PoW for both roles? Both applications ask the prover to solve a puzzle, suggesting that a single PoW might serve both roles. A verifiable delay function (VDF) [BBBF18] would suffice: it provides a sequential delay and a computationally unique output per puzzle, with efficient verification. A circuit that evaluates the hash at q_C gates makes at most q_C sequential oracle calls, so a delay requirement $D > q_C$ prevents the circuit from computing the challenge. Computational uniqueness rules out finding a second accepted output for the

same puzzle, even with polynomially more work than an honest evaluation, which means a malicious prover cannot obtain a different challenge by changing the output. However, VDFs cannot be constructed in the ROM without additional assumptions [MSW20; GRY25; AAKTC26].

A natural question is whether a weaker primitive in the ROM can serve both applications. XFS relies on strong soundness: a prover that performs substantially less online work than the honest prover solves a random puzzle with only negligible probability, even after bounded preprocessing. This is weaker than sequentiality, since the number of sequential query rounds is at most the total number of queries. For grinding, on the other hand, we observe that it suffices to require *non-amortization*: producing k distinct accepted puzzle–solution–proof triples must cost, in expectation, roughly k times as much as producing one. Non-amortization is much weaker than computational uniqueness, since it allows multiple solutions to the same puzzle as long as each requires additional queries.

Neither canonical construction satisfies both properties. Preimage PoW [Bac02; Nak08] is non-amortizing but not strong: solving a puzzle costs D queries, yet even one fresh query succeeds with probability $1/D$, which is non-negligible for polynomial D . The Merkle-tree PoW of Arnon and Yagev [AY25] is strong against polynomially bounded preprocessing, but amortizable. When the tree has many more leaves than sampled openings, a prover that has built one tree for work D can, with high probability, obtain a second solution to the same puzzle using only $O(\log D)$ fresh queries.

We therefore ask:

Does there exist a PoW in the pure random oracle model that is both strong and non-amortizing?

1.1 Our results

We prove an impossibility result for a single PoW and give a construction using two PoWs.

1.1.1 Impossibility of a single PoW

Our first result answers the question above in the negative.

Theorem 1 (Informal). *No PoW in the ROM with efficient verification is both strong and non-amortizing.*

Here efficient verification means that the verifier makes significantly fewer than $\sqrt[3]{D}$ queries.

Our proof consists of two reductions. The first transforms a strong, non-amortizing PoW into a PoW that is strong and computationally unique, i.e., finding a second accepted solution–proof pair is hard. The second transforms a strong, computationally unique PoW into one that is strong and statistically unique. Both transformations preserve strong soundness and verifier query complexity. We then apply the search bound of [GRY25], which says a statistically unique search problem can be solved with polynomially many queries relative to the verifier’s query complexity. Since the verifier is efficient, the transformed PoW can be solved with far fewer than D queries, contradicting strong soundness.

As a corollary, the second transformation extends the bound of [GRY25], which builds on Smyth’s work [Smy02], from statistically unique to computationally unique search problems, resolving an open problem of [GRY25].

Unique proofs. In this paper, we consider Fiat–Shamir transformations in which the hash absorbs everything the prover sends, including the PoW solution and its proof. This is why non-amortization counts distinct solution–proof pairs rather than distinct solutions: if two distinct proofs for the same puzzle–solution pair are easy to find, a prover can fix one solution, vary the proof, and obtain a fresh challenge for each proof at negligible cost. In particular, by definition no PoW *without* computationally unique proofs is non-amortizing,

and for PoWs with unique proofs the two counts coincide. Every PoW we are aware of has unique proofs. Preimage PoW [Bac02; Nak08], Equihash [BK16], and Cuckoo Cycle [Tro15] are proofless: the solution is a nonce or a set of indices, and verification recomputes hashes. In the Merkle-tree PoW of [AY25] and in proofs of sequential work [MMV13; CP18], the solution is a Merkle root and the proof consists of openings at positions derived by hashing the puzzle and root, which are computationally unique.

One might instead hash only the solution and leave the proof out of the hash. This would weaken the requirement to non-amortization of solutions, as the output of a VDF certifies work on its own. We absorb the proof as well, for two reasons. Absorbing every prover message is the standard practice, and omitting messages from the hash has repeatedly caused attacks on deployed systems [BPW12; HLPT20; DMWG23; CPV24; NHB24], including on a VDF [DMWG23]. And the choice costs nothing for the PoWs we know of. The PoW used for grinding in practice [Sta21; Pol22; DP26; BRW26] is a proofless nonce, and the schemes above are proofless or have computationally unique proofs, so the two notions coincide. That said, without unique proofs, Theorem 1 says nothing about non-amortization of solutions, and whether such a PoW can be both strong and non-amortizing in solutions remains open.

1.1.2 XFS with grinding

Since a single PoW cannot serve both roles, we use one for each. The prover first solves the strong PoW, uses its solution and proof to define the puzzle for the non-amortizing PoW, and then solves that puzzle. The FS challenge is derived by hashing both solution–proof pairs. The order matters: the non-amortizing PoW must be solved second, since otherwise the adversary could grind on the strong PoW without solving the non-amortizing one.

Theorem 2 (Informal). *XFS-with-grinding compiles any relativized Σ -protocol with knowledge error κ conditioned on the pre-challenge view into a non-interactive argument of knowledge in the relativized ROM. Let the strong PoW have difficulty D_A and the non-amortizing PoW have difficulty D_B . Every efficient adversary has knowledge error at most*

$$\frac{q_B}{D_B} \cdot \kappa + \text{negl}(\lambda),$$

and the honest prover makes $D_A + D_B$ queries beyond those of the Σ -protocol prover.

The non-amortization PoW bounds the number of distinct accepted puzzle–solution–proof triples, and hence candidate challenges, by about q_B/D_B (see Definition 3.12). The strong PoW rules out diagonalization attacks and contributes only a negligible term.

1.2 Related work

Impossibility results in the ROM. Our impossibility result is closely related to the barriers to constructing VDFs in the ROM. A VDF requires a delay that cannot be substantially shortened by parallel computation. Mahmoody, Smith, and Wu [MSW20] ruled out VDFs with perfect uniqueness and perfect completeness. Under perfect uniqueness, each puzzle has at most one accepted solution for every oracle. Later work extended the impossibility to computationally unique VDFs [GRY25]. [AAKTC26] later extends the impossibility to private-coin setup [AAKTC26]. This weaker notion of uniqueness allows alternative solutions to exist as long as they are hard to find.

The connection to our result comes from a search bound of [GRY25] that does not require a sequential delay. Building on [Smy02; KSS11], they bound the number of queries needed to solve a statistically unique search problem in terms of the verifier’s query complexity. Statistical uniqueness means that multiple

accepted solutions exist only with negligible probability over the random oracle. They ask whether an analogous bound holds under computational uniqueness. We extend the bound through a reduction from computational to statistical uniqueness. This addresses their question for search problems with no setup or auxiliary proofs, over any finite response alphabet, subject to the stated error bounds. We also require computational uniqueness to hold against adversaries allowed as many queries as the honest solver.

A separate barrier concerns the relativized Σ -protocols our transformation takes as input: Barbara, Chiesa, and Guan [BCG25] rule out succinct relativized arguments for general oracle computations in the ROM. Our XFS-with-grinding construction, like XFS [AY25], is a conditional transformation that assumes a suitable relativized Σ -protocol is available.

Proofs of work and grinding. Proof of work [DN93] requires computation to deter abuse. The most widely deployed construction is based on finding a preimage of a hash function that meets a target condition [Bac02; Nak08]. This construction is non-amortizing. Difficulty notions for solving many puzzles at once appear in [DGN03; DNW05; SKRBG11], and Ball et al. [BRSV18] define non-amortizability for PoWs from worst-case assumptions. Hash-based proof systems such as ethSTARK [Sta21], Plonky2 [Pol22, Sections 6.2 and 7.3], and Flock [BRW26] use PoW for challenge grinding. In critical rounds, the prover solves a transcript-bound PoW before deriving the corresponding Fiat–Shamir challenge. This limits the number of challenges an adversary can try. In the ROM, d bits of grinding cost $D = 2^d$ hash trials and multiply the corresponding round-by-round soundness error [Can+19] by $1/D$; the remaining rounds and hash-collision terms retain their contributions [Sta21; BT24]. At a fixed security target, this can reduce the number of opening queries, and hence the proof size and verifier time in the final compiled SNARK. It has also been used to achieve $2^{-\lambda}$ soundness error in a field of size $\leq 2^\lambda$ [BRW26], since such fields are efficient in practice. Under its stated hypotheses, Theorem 2 combines this soundness amplification with security against diagonalization attacks in the relativized ROM of XFS.

Attacks on Fiat–Shamir and defenses. Fiat–Shamir [FS87] is proven secure in the ROM [BR93; PS96], but, as discussed earlier, ROM security need not survive instantiating the oracle with a concrete hash [CGH98; GK03]. Bartusek et al. [BBHMR19] broke the transformation of Kilian’s argument [Kil92] with a contrived hash, and Khovratovich et al. [KRS25] gave the practical diagonalization attack on a GKR-based argument described in Section 1, which Fenzi [Fen26] extended to proof systems whose instance is generated by an adversarial program. Their countermeasure, restricting the depth or degree of the circuit so that it cannot compute the hash, has neither a security proof nor a verifier succinct in the circuit depth. XFS [AY25], also discussed in Section 1, obtains a conditional security theorem in a relativized ROM from a strong PoW and gives a Merkle-tree PoW construction. The full version of [AY25, Section 2.4] explicitly states that its theorem does not directly prove security for the GKR-based protocol. The December 2025 full version of [KRS25, Section 5] questions whether this model captures its attack and leaves concrete security of the proposed mitigations open. Our theorem inherits this model limitation. By Theorem 1, no strong PoW, theirs included, is non-amortizing in the parameter regime outlined in Section 2.1. In Theorem 2, grinding is restored by using two PoWs.

A different line of work constructs *correlation-intractable* hash families, a notion introduced in [CGH98], and uses them to instantiate Fiat–Shamir in the standard model [KRR17; CCRR18; HL18; Can+19; PS19; BKM20; KLV23; CGJJZ23; LV20; JKKZ21; CJJ22; HLR21]. These specialized constructions rely on assumptions such as LWE, DDH, or obfuscation and support different classes of protocols, yielding NIZKs for NP and succinct arguments for classes including P and Batch-NP. The cited constructions without obfuscation do not give a general-purpose publicly verifiable SNARG for all NP. XFS and our work use an efficient hash function, modeled as a random oracle, together with PoW to address diagonalization within the relativized model.

2 Technical overview

In Section 2.1, we review the search bound of [GRY25] and explain the two reductions that transform a strong and non-amortizing PoW into a search problem to which that bound applies. In Section 2.2, we describe XFS-with-grinding and show how strong soundness and non-amortization are used in its security proof.

PoWs as search problems. We measure work in random oracle queries; local computation is free, so no computational assumptions are needed. Recall that D denotes the PoW’s difficulty, the expected work for solving one puzzle. For this overview, assume for ease of discussion that the honest prover makes at most D queries. Let $q_v \geq 1$ bound the verifier’s number of queries. We are interested in the regime $q_v \ll D$, where verification is polynomially cheaper than solving. An output of the PoW is a solution s . A unique proof, if any, is folded into the solution (Remark 3.9).

For a fixed puzzle z , a PoW is a *search problem* over the random oracle f : the solver P is the honest prover on z , the verifier V is the PoW verifier on z , and an *accepted output* on f is a solution s with $V^f(s) = 1$. Both P and V are deterministic, so the oracle is the only source of randomness. We state the properties we need in this language; the corresponding PoW property is the same statement averaged over a random puzzle.

The search problem is *correct* if V accepts P ’s output on all but a negligible fraction of oracles. It is *strong* if every adversary making far fewer than D queries finds an accepted output with negligible probability; to rule out a strong PoW, it therefore suffices to solve its search problem for a random puzzle with far fewer than D queries. It is *computationally unique* for a given query budget (which we set to be D below) if every adversary within that budget finds an accepted output different from P ’s with only negligible probability, and it is *statistically unique* if a random oracle admits an accepted output different from P ’s only with negligible probability. Note that a correct and statistically unique search problem has exactly one accepted output on all but a negligible fraction of oracles.

As in Section 1, the PoW is *non-amortizing* if every adversary making at most q_A queries in expectation finds at most q_A/D distinct accepted puzzle–solution pairs in expectation, up to a negligible additive term γ . For simplicity of exposition, in this section we restrict the impossibility argument to the *strict-time* variant of this notion: fix a small constant $\eta > 0$; for every $k > 1$, every adversary making at most $k(1 - \eta)D$ queries finds k distinct accepted pairs with only negligible probability. The expected-time version of the argument follows the same route using expected query costs; we point to it at the end of Section 2.1.

2.1 Impossibility of a strong and non-amortizing PoW

Review: statistically unique search problems. Consider a correct and statistically unique search problem, and fix a sufficiently small constant $\delta > 0$ bounding the fraction of oracles on which there is not exactly one accepted output. [GRY25] show that an adversary can find an accepted output with $O(q_v^2/\delta)$ queries and probability at least $1 - 6\sqrt{\delta}$. Since δ is a constant, the query complexity is $O(q_v^2)$, so the bound rules out strong soundness whenever $q_v^2 \ll D$.

We first consider the case where *every* oracle has exactly one accepted output. Call a complete accepting verifier transcript a *certificate*: it is a partial assignment to at most q_v positions of f , and any oracle that agrees with it makes V accept the corresponding output. In this case, certificates for different outputs are pairwise inconsistent: if two of them agreed on their shared positions, there exists some oracle that is consistent with both, and V would accept two different outputs on it. Distinct accepting transcripts for the same output are also inconsistent, since the verifier is deterministic. The classical argument bounding decision tree complexity by the square of certificate complexity then gives an $O(q_v^2)$ -query algorithm. The algorithm repeatedly picks any certificate consistent with the oracle answers it has seen so far and queries all of its positions. If all

queried answers agree with the chosen certificate, the algorithm returns its corresponding output. Otherwise, the chosen certificate disagrees with the actual accepting transcript at a previously unqueried position. Since the true certificate has at most q_v positions, at most q_v rounds of at most q_v queries each suffice.

The argument breaks down as soon as uniqueness is violated on a small fraction of oracles: certificates for different outputs may then overlap, and a round may reveal nothing new about the true certificate. Kahn, Saks, and Smyth [KSS11] proved a conjecture of Rudich [Rud88] about families of certificates that cover almost every input uniquely. Building on this result, Smyth [Smy02] established an approximately quadratic bound for Boolean functions, resolving a conjecture of Tardos. [GRY25] extended this bound to search problems with arbitrarily many outputs.

Why the search bound does not apply directly. The search bound assumes statistical uniqueness, but neither strong soundness nor non-amortization alone implies it. Non-amortization does not restrict how many outputs the verifier accepts for a puzzle; it only asks that each additional output, including one obtained by modifying an existing tree, require roughly as much additional work as computing the output of a fresh puzzle. Computational uniqueness is closer to what we need, but it also allows multiple accepted outputs. It only says that an adversary with bounded queries cannot find an accepted output different from the honest prover's. In fact, [GRY25] explicitly asked whether the search bound extends from statistical uniqueness to computational uniqueness.

Our proof has two reductions. Reduction 1 uses non-amortization to obtain a search problem that is strong and computationally unique. Reduction 2 turns computationally unique search problems into statistically unique search problems. Both reductions preserve strong soundness and the verifier's query complexity q_v , which allows us to apply the search bound from [GRY25] at the end.

Reduction 1: non-amortizing implies computationally unique. We construct a search problem with computational uniqueness against adversaries whose query budget matches that of the honest solver. Let h be a number of queries that is a small fraction of D ; we fix it below. Write $D_{\text{alt}} = D - h$. We define the *residual search problem* as follows.

1. *Puzzle generator.* Sample a PoW puzzle and run the honest prover for its first h queries. The new puzzle consists of the original puzzle, the saved prover state, and the queries and answers seen so far.
2. *Solver.* Continue the honest prover from the saved state to obtain its solution. This requires at most D_{alt} further queries.
3. *Verifier.* Check the proposed solution using the original PoW verifier on the original puzzle. This requires at most q_v queries.

We claim that the residual problem is computationally unique against D_{alt} -query adversaries. Suppose a D_{alt} -query adversary finds an accepted output s' that differs from the residual solver's output s with non-negligible probability. We can produce both by running the prefix once, then running the honest solver and the adversary from the saved state, for a total of $h + 2D_{\text{alt}} = 2D - h$ queries. This is an amortization attack on the original PoW: two distinct accepted solutions for the same puzzle with fewer queries than two honest runs. By non-amortization, an adversary making at most $2(1 - \eta)D$ queries produces two such solutions with only negligible probability. Choosing h slightly larger than $2\eta \cdot D$ gives

$$h + 2D_{\text{alt}} = 2D - h < 2(1 - \eta)D,$$

so the attack falls within the budget of non-amortization, a contradiction. Sharing the prefix between the two runs is what gives computational uniqueness against adversaries with the same query budget as the residual solver. Without the prefix, the honest run costs D queries, so non-amortization only gives computational uniqueness against adversaries making fewer than $(1 - 2\eta)D$ queries.

Strong soundness is preserved as well. Any solver for the residual problem gives a solver for the original PoW: run the honest prover for h queries to produce the saved state, then run the residual solver. The strong-soundness bound therefore transfers with an additive cost of h queries.

Reduction 2: computationally unique implies statistically unique. Consider a search problem with a D-query solver P and a q_v -query verifier V . We assume computational uniqueness against D-query adversaries.

We replace V by a more restrictive verifier $\hat{V}$ so that alternative accepted outputs rarely exist, while preserving correctness. If $\hat{V}$ accepts an output different from P 's on a non-negligible fraction of oracles, we show how to find such an output with D queries, which would contradict computational uniqueness. Thus, $\hat{V}$ is statistically unique.

An accepting certificate for an output fixes only the oracle answers that V needs, whereas the solver P typically reads many other positions. Changing those other positions may change P 's output, but the oracle still agrees with the certificate, so V still accepts the original output. When this happens, the original output is an alternative to P 's output on the changed oracle. Fix a resampling probability $\theta \in (0, 1)$. This gives the following θ -resampling adversary.

1. Initialize an empty table for a lazily sampled oracle g .
2. Run P with oracle g . For each new query x , independently set $g(x)$ to a uniform oracle output with probability θ , and to $f(x)$ otherwise, querying the real oracle f only in the latter case. Store the answer and return it whenever P queries x .
3. Output the solution s produced by P .

We analyze the probability that the adversary's output is accepted by V on f but differs from the honest solver's output on f . The adversary makes at most D queries to the real oracle, since it runs P once, so a non-negligible success probability would contradict computational uniqueness.

The oracle g is a θ -resampling of f , sampled only at the positions queried by P in its run on g . The joint distribution of f and g is symmetric: we may instead sample g as a random oracle, then obtain f by independently resampling each position with probability θ .

Suppose P produces an accepted output on g , and consider its certificate (the oracle queries and answers from V 's accepting run on this output with oracle g). Since the certificate fixes at most q_v positions, with probability at least $(1 - \theta)^{q_v}$ none of these positions is resampled. In this case, V also accepts the same output on f , while the positions outside the certificate are still independently resampled with probability θ . The attack succeeds if these changes cause P to produce a different output on f .

We define the *certificate disagreement* Δ as follows. Sample a random oracle g and let $s = P^g$. If V accepts this output on g , sample an oracle g' that agrees with g on the accepting certificate and has independent uniform values at all other positions. If V rejects, set $g' = g$. Then

$$\Delta := \Pr \left[V^g(s) = 1 \wedge P^{g'} \neq s \right].$$

The probability is over g and the sampling of g' , with rejected runs contributing zero.

Let Δ_θ be the probability of changing P 's output in the same experiment, except that each position outside the certificate is resampled independently with probability θ . The certificate is still fixed, and rejected runs still contribute zero. In particular, $\Delta_1 = \Delta$. The main step in our resampling analysis is to prove $\Delta_\theta \geq \theta \Delta$ (see the proof of Lemma 5.6).

Combining this with the $(1 - \theta)^{q_v}$ bound above gives

$$\Pr [\text{the adversary succeeds}] \geq (1 - \theta)^{q_v} \Delta_\theta \geq \theta (1 - \theta)^{q_v} \Delta.$$

Choosing $\theta = 1/(q_v + 1)$ gives success probability $\Omega(\Delta/q_v)$. If Δ were non-negligible, this D-query adversary would contradict computational uniqueness. Hence Δ is negligible.

We define $\widehat{V}$ as follows. On input s , it runs V and rejects if V rejects. Otherwise, it accepts exactly when the resulting certificate τ is *saturated*, meaning that for a uniformly random oracle g' ,

$$\Pr_{g'} [P^{g'} = s \mid g' \text{ agrees with } \tau] \geq \frac{1}{2}.$$

To prove correctness, observe that an accepted honest output is rejected by $\widehat{V}$ only if its certificate is not saturated. In that case, resampling outside the certificate changes P 's output with probability greater than $1/2$. By the definition of Δ and Markov's inequality,

$$\Pr_g [V^g(P^g) = 1 \wedge \widehat{V}^g(P^g) = 0] \leq 2\Delta.$$

Since $\widehat{V}$ rejects whenever V rejects, its rejection event splits into two disjoint cases: V rejects, or V accepts but $\widehat{V}$ rejects. Hence,

$$\Pr_g [\widehat{V}^g(P^g) = 0] = \Pr_g [V^g(P^g) = 0] + \Pr_g [V^g(P^g) = 1 \wedge \widehat{V}^g(P^g) = 0] \leq \Pr_g [V^g(P^g) = 0] + 2\Delta.$$

Thus, the correctness error increases by at most 2Δ .

For statistical uniqueness, consider the oracles that agree with a saturated certificate but make P produce a different output. Since the certificate is saturated, the probability of these oracles is at most twice that certificate's contribution to Δ . If $\widehat{V}$ accepts two distinct outputs on the same oracle, at least one differs from P 's output. Summing over saturated certificates therefore gives

$$\Pr_g [\widehat{V}^g \text{ accepts two distinct outputs}] \leq 2\Delta.$$

An accepted output different from P 's can exist only if $\widehat{V}$ rejects the honest output or accepts at least two outputs. Combining the two bounds gives

$$\Pr_g [\exists s \neq P^g : \widehat{V}^g(s) = 1] \leq \Pr_g [\widehat{V}^g(P^g) = 0] + 2\Delta \leq \Pr_g [V^g(P^g) = 0] + 4\Delta.$$

Checking whether a certificate is saturated requires only local computation, which is free in our query model, so $\widehat{V}$ makes at most as many oracle queries as V . Every output accepted by $\widehat{V}$ is also accepted by V , so the transformation preserves strong soundness.

Putting it together. To prove Theorem 1, we first spend h queries to generate the residual puzzle. Its solver uses at most $D_{\text{alt}} = D - h$ queries, and Reduction 1 gives computational uniqueness against adversaries with the same budget. Reduction 2 therefore applies with D_{alt} in place of D , giving a correct and statistically unique search problem with a q_v -query verifier. The bound of [GRY25] then gives an $O(q_v^2)$ -query solver with constant success probability. Its output is also accepted by the original PoW verifier. The total cost is $h + O(q_v^2)$ queries, which contradicts strong soundness when h is a small fraction of D and $q_v^2 \ll D$. For non-amortization in expectation, the honest prover has cutoff q_p and expected query cost at most D ; we run both reductions and the search bound using expected query budgets (Sections 4 and 6).

2.2 XFS with grinding

To prove Theorem 2, we sequentially compose a strong PoW PoW_A and a non-amortizing PoW PoW_B . Write D_A and D_B for their difficulties, measured in honest prover queries. The second puzzle includes the first solution, so that changing either component also requires fresh work for PoW_B , even if PoW_A is amortizable.

We apply this construction to a relativized Σ -protocol Π , in which the circuit being proven may query the same random oracle as the prover and verifier. Write m_1 , ρ , and m_2 for its first message, challenge, and response. The prover computes ρ as follows.

1. Hash the circuit, instance, and m_1 to obtain a puzzle for PoW_A , and solve it.
2. Hash the same values together with the first solution to obtain a puzzle for PoW_B , and solve it.
3. Hash the circuit, instance, m_1 , and both solutions to obtain ρ .

The prover then computes m_2 using ρ , and the verifier checks both PoW solutions and the Σ -protocol transcript. The honest prover's expected PoW cost is $D_A + D_B$.

A diagonalization attack makes the circuit defining the statement compute the FS challenge ρ for a proof about that same circuit. The oracle query defining ρ for an accepted proof includes a valid solution to PoW_A . Following the XFS reduction of [AY25], we extract this solution by evaluating the circuit and inspecting its oracle queries. We choose D_A larger by a constant factor than the query cost of evaluating the circuit and checking the candidate solutions. Strong soundness then implies that this extraction, and hence the attack, succeeds with only negligible probability, even after bounded preprocessing.

Recall that a prover can try different inputs to the challenge hash and choose one that gives a favorable challenge. Grinding uses PoW to require additional work for each such attempt. We show that reusing a PoW_B puzzle–solution pair does not give the adversary another challenge. Suppose two valid challenge queries use the same pair. Unless an oracle collision occurs, the two queries must contain the same circuit digest, instance, m_1 , and first solution, since these values determine the PoW_B puzzle. The challenge queries therefore coincide. Thus, each new valid attempt requires a distinct accepted pair for PoW_B . Non-amortization bounds the expected number of these pairs by $q_B/D_B + \gamma_B$, where γ_B is the additive term in the non-amortization of PoW_B and q_B includes verification and transcript completion. If Π has knowledge error κ conditioned on the view before each fresh challenge, then summing over these attempts gives a grinding term of $(q_B/D_B + \gamma_B) \cdot \kappa$. For preimage PoW, $\gamma_B = 0$.

3 Preliminaries

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 $f : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$.*

Definition 3.2 (Almost-sure halting). *An algorithm **halts almost surely** in an experiment if it terminates after finitely many computation steps with probability 1, over all randomness in the experiment, including the random oracle. Nontermination is allowed only on an event of probability zero. Finite expected query count alone does not guarantee this condition, since an algorithm may loop forever without making further queries.*

3.1 Relativized Σ -protocol

We adapt the following definitions from [AY25]. The indexer is deterministic.

Definition 3.3. *A triplet of algorithms $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ is a relativized SP in the ROM for a family of Boolean circuits $\mathcal{C}$ if the following properties hold:*

- **Completeness.** *For every $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}$, instance $\mathbb{x}$, witness $\mathbb{w}$, and oracle $f \in \mathcal{O}(\lambda)$ for which $C^f(\mathbb{x}, \mathbb{w}) = 1$, it holds that*

$$\Pr \left[\mathbf{V}^f(\psi, \mathbb{x}, m_1, \rho, m_2) = 1 \mid \begin{array}{l} (m_1, \text{aux}) \leftarrow \mathbf{P}^f(C, \mathbb{x}, \mathbb{w}) \\ \rho \leftarrow \{0, 1\}^\lambda \\ m_2 \leftarrow \mathbf{P}^f(\text{aux}, \rho) \\ \psi \leftarrow \mathbf{I}^f(C) \end{array} \right] = 1.$$

- **Adaptive (straightline) knowledge.** *There exists a deterministic polynomial-time extractor $\mathbf{E}$ such that the following holds for every $\lambda, q \in \mathbb{N}$, malicious q -query SP prover $\tilde{\mathbf{P}}$, circuit C^* , instance $\mathbb{x}^*$, first message m_1^* , prover state aux^* , and first-phase query-answer transcript tr_1^* . If the **pre-challenge view** $v^* := (C^*, \mathbb{x}^*, m_1^*, \text{aux}^*, \text{tr}_1^*)$ satisfies*

$$\Pr \left[\text{view}_1 = v^* \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, m_1, \text{aux}) \xleftarrow{\text{tr}_1} \tilde{\mathbf{P}}^f \\ \text{view}_1 := (C, \mathbb{x}, m_1, \text{aux}, \text{tr}_1) \end{array} \right] > 0,$$

then

$$\Pr \left[\begin{array}{l} C \in \mathcal{C} \\ \wedge C^f(\mathbb{x}, \mathbb{w}) = 0 \\ \wedge \mathbf{V}^f(\psi, \mathbb{x}, m_1, \rho, m_2) = 1 \\ \text{conditioned on} \\ \text{view}_1 = v^* \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, m_1, \text{aux}) \xleftarrow{\text{tr}_1} \tilde{\mathbf{P}}^f \\ \text{view}_1 := (C, \mathbb{x}, m_1, \text{aux}, \text{tr}_1) \\ \rho \leftarrow \{0, 1\}^\lambda \\ m_2 \leftarrow \tilde{\mathbf{P}}^f(\text{aux}, \rho) \\ \mathbb{w} \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_1) \\ \psi \leftarrow \mathbf{I}^f(C) \end{array} \right] \leq \kappa(\lambda, q).$$

- **Adaptive transcript binding.** *The same extractor $\mathbf{E}$ satisfies the following for every $\lambda, q \in \mathbb{N}$, malicious q -query SP prover $\tilde{\mathbf{P}}$, and q -query algorithm $\mathbf{A}$,*

$$\Pr \left[\begin{array}{l} C \in \mathcal{C} \\ \wedge \mathbb{w} \neq \mathbb{w}' \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, m_1, \text{aux}) \xleftarrow{\text{tr}_1} \tilde{\mathbf{P}}^f \\ \perp \xleftarrow{\text{tr}_2} \mathbf{A}^f(\text{aux}) \\ \mathbb{w} \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_1) \\ \mathbb{w}' \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_1 \parallel \text{tr}_2) \end{array} \right] \leq \kappa_{\text{tb}}(\lambda, q).$$

- **Adaptive (straightline) indexer-knowledge.** There exists a deterministic polynomial-time extractor $\mathbf{E}_I$ such that for every $\lambda, q \in \mathbb{N}$ and malicious q -query adversary $\mathbf{A}$,

$$\Pr \left[\begin{array}{l} C \neq C' \\ \wedge \mathbf{I}^f(C) = \psi \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (\psi, \text{aux}) \xleftarrow{\text{tr}} \mathbf{A}^f \\ C' \leftarrow \mathbf{E}_I(\psi, \text{tr}) \\ C \leftarrow \mathbf{A}^f(\text{aux}) \end{array} \right] \leq \kappa_I(\lambda, q).$$

Definition 3.4 (Relativized NARKs in the ROM). A triplet of algorithms $(\mathcal{I}, \mathcal{P}, \mathcal{V})$ in the random oracle model is a relativized NARK in the ROM for a family of Boolean circuits $\mathcal{C}$ if the following properties hold:

- **Completeness.** For every $\lambda, q \in \mathbb{N}$, circuit $C \in \mathcal{C}$, instance $\mathbb{x}$, and witness $\mathbb{w}$, the honest prover $\mathcal{P}$, making at most q queries, satisfies

$$\Pr \left[\begin{array}{l} C^f(\mathbb{x}, \mathbb{w}) = 1 \\ \wedge \mathcal{V}^f(\psi, \mathbb{x}, \pi) \neq 1 \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \pi \leftarrow \mathcal{P}^f(C, \mathbb{x}, \mathbb{w}) \\ \psi \leftarrow \mathcal{I}^f(C) \end{array} \right] \leq \alpha_{\text{ARG}}(\lambda, q).$$

- **Adaptive (straightline) knowledge.** There exists a $q_{\mathcal{E}}$ -query argument extractor $\mathcal{E}$ such that the following holds for every $\lambda, q \in \mathbb{N}$ and malicious q -query argument prover $\tilde{\mathcal{P}}$, where $q_{\mathcal{E}} = q_{\mathcal{E}}(\lambda, q)$,

$$\Pr \left[\begin{array}{l} C \in \mathcal{C} \\ \wedge C^f(\mathbb{x}, \mathbb{w}) = 0 \\ \wedge \mathcal{V}^f(\psi, \mathbb{x}, \pi) = 1 \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, \pi) \xleftarrow{\text{tr}_{\tilde{\mathcal{P}}}} \tilde{\mathcal{P}}^f \\ \mathbb{w} \leftarrow \mathcal{E}^f(C, \mathbb{x}, \pi, \text{tr}_{\tilde{\mathcal{P}}}) \\ \psi \leftarrow \mathcal{I}^f(C) \end{array} \right] \leq \kappa_{\text{ARG}}(\lambda, q).$$

3.2 Prefix-avoiding padder

The following definition and lemma on prefix-avoiding padders are from [AY25].

Definition 3.5 ([AY25, Definition 5.1]). An oracle algorithm $\mathbf{M}$ is ε_{pad} -prefix-avoiding with respect to deterministic padder Padder if for every $\lambda, q \in \mathbb{N}$ and q -query adversary $\mathbf{A}$,

$$\Pr \left[(\tau, \gamma) \in \text{tr} \cup \text{tr}_{\mathbf{M}} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (\gamma, m) \leftarrow \mathbf{A}^f \\ \tau \xleftarrow{\text{tr}} \text{Padder}^f(\gamma) \\ y \xleftarrow{\text{tr}_{\mathbf{M}}} \mathbf{M}^f(m) \end{array} \right] \leq \varepsilon_{\text{pad}}(\lambda, q).$$

Lemma 3.6 ([AY25, Lemma 5.2]). Let $\mathbf{M}$ be an oracle algorithm that makes at most q queries and has code description $\langle \mathbf{M} \rangle$. Then the following holds:

1. If every query made by $\mathbf{M}$ has length at most ℓ , then $\mathbf{M}$ is 0-prefix-avoiding with respect to the padder $\text{Padder}^f(\gamma) = 0^{\ell+1}$.
2. Let $\mathbf{M}$ be an algorithm that gets ℓ bits as input. Then, $\mathbf{M}$ is $O(q/2^{2\lambda})$ -prefix-avoiding with respect to the padder $\text{Padder}^f(\gamma) = (f(\langle \mathbf{M} \rangle, \gamma, 1), \dots, f(\langle \mathbf{M} \rangle, \gamma, k))$, where k satisfies $k \cdot \lambda \geq \ell + 2\lambda$.

Remark 3.7. Our application requires prefix avoidance for the fixed verifier $\mathbf{V}$, as in [AY25]. As discussed in [AY25, Remark 5.4], a concrete verifier may leave a known oracle region unused for every choice of prover messages. The pad can then be fixed in advance to place the Fiat–Shamir challenge queries in that region.

3.3 Proof of work

Definition 3.8 (Proof of work (PoW) in the ROM). A **proof of work PoW in the ROM** is a tuple of oracle-aided algorithms $\text{PoW} = (\text{Setup}, P, V)$ such that for every λ, D and $f \in \mathcal{O}(\lambda)$, the following hold:

- $\text{Setup}^f(1^\lambda, D) \rightarrow \text{pp}$: On input the security parameter λ and the expected total number of queries D for P to solve a puzzle, the **deterministic** setup algorithm Setup outputs the public parameters pp , where pp determines a puzzle space $\mathcal{Z}$ and a solution space $\mathcal{S}$.
- $P^f(\text{pp}, z) \rightarrow s$: On input the public parameter pp and a puzzle $z \in \mathcal{Z}$, the **deterministic** proving algorithm P outputs a solution s .
- $V^f(\text{pp}, z, s) \rightarrow \{0, 1\}$: On input the public parameters pp , a puzzle $z \in \mathcal{Z}$, and a solution $s \in \mathcal{S}$, the **deterministic** verification algorithm V outputs a bit indicating whether it accepts or rejects.

We require that Setup and V make at most q_{Setup} and q_V queries, respectively, to the random oracle, where $q_{\text{Setup}} = q_{\text{Setup}}(\lambda, D)$ and $q_V = q_V(\lambda, D)$. In practice, we want $q_{\text{Setup}} \ll D$ and $q_V \ll D$.

We consider PoW together with a specified randomized oracle-aided puzzle generator $\text{Gen}^f(\text{pp}) \rightarrow z \in \mathcal{Z}$ that makes at most q_{Gen} queries. Correctness, success, soundness, and uniqueness are defined relative to this specified generator. For simplicity, we omit dependence on the generator in the error functions.

Remark 3.9 (Proofs). As discussed in Section 1.1.1, we consider PoWs whose proofs are computationally unique, and the Fiat–Shamir hash absorbs the solution together with its proof. Formally, this is equivalent to a PoW without a separate proof: a unique proof can be folded into the solution, and every notion below then refers to the combined output. For notational clarity, we therefore omit the proof and write s for the output.

PoWs have the following properties. In these properties, D is the expected query count of the honest prover, and q_P is a query cutoff. The prover returns a failure output $\perp$ if its next query would exceed this cutoff. The correctness error $\alpha(\lambda, D, q_P)$ includes failures caused by this cutoff. Failure outputs are rejected without oracle queries.

Definition 3.10 (Correctness). $\text{PoW} = (\text{Setup}, P, V)$ has correctness error α relative to its specified generator Gen if, for every $\lambda, D, q_P \in \mathbb{N}$, the honest prover P with cutoff q_P satisfies

$$\Pr \left[V^f(\text{pp}, z, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow P^f(\text{pp}, z) \end{array} \right] \leq \alpha(\lambda, D, q_P).$$

When $\alpha = 0$, we say PoW has **perfect correctness**.

Definition 3.11 (Strong Soundness). $\text{PoW} = (\text{Setup}, P, V)$ has soundness error ϵ relative to its specified generator Gen if, for every $\lambda, D, q_{A_1}, q_{A_2} \in \mathbb{N}$, every adversary $A = (A_1, A_2)$ such that A_1 makes at most q_{A_1} queries and outputs an auxiliary string aux , and A_2 makes at most q_{A_2} online queries satisfies

$$\Pr \left[V^f(\text{pp}, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \text{aux} \leftarrow A_1^f(\text{pp}) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A_2^f(\text{aux}, z) \end{array} \right] \leq \epsilon(\lambda, D, q_{A_1}, q_{A_2}).$$

Definition 3.12 (*k*-fold non-amortization). For $\lambda, D, k, q_A \in \mathbb{N}$ with $\lambda, D, k \geq 1$, and $\gamma \geq 0$, the PoW has *k*-fold non-amortization with parameter γ if every adversary A that halts almost surely, makes at most q_A oracle queries in expectation, and outputs at most k pairs (If the adversary outputs fewer than k pairs, pad with $(\perp, \perp)$.) after verifying each of them satisfies

$$\mathbb{E} \left[N \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \{(z^{(i)}, s^{(i)})\}_{i=1}^k \leftarrow A^f(\text{pp}) \end{array} \right] \leq \frac{q_A}{D} + \gamma(\lambda, D, q_A). \quad (1)$$

where N counts distinct accepted puzzle–solution pairs:

$$N := \left| \left\{ (z^{(i)}, s^{(i)}) \mid \begin{array}{l} i \in [k], \\ V^f(\text{pp}, z^{(i)}, s^{(i)}) = 1 \end{array} \right\} \right|.$$

The computational uniqueness notion below uses a strict query bound.

Definition 3.13 (Computational uniqueness). PoW = (Setup, P, V) has computational uniqueness error ζ relative to its specified generator Gen if for every $\lambda, D \in \mathbb{N}$, $q_A = \text{poly}(\lambda, D)$, and a q_A -query adversary A ,

$$\Pr \left[\begin{array}{l} s \neq s' \\ \wedge V^f(\text{pp}, z, s) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A^f(\text{pp}, z) \\ s' \leftarrow P^f(\text{pp}, z) \end{array} \right] \leq \zeta(\lambda, D, q_A).$$

Definition 3.14 (Statistical uniqueness). PoW = (Setup, P, V) has statistical uniqueness error ζ^* relative to its specified generator Gen if for every $\lambda, D \in \mathbb{N}$, and an unbounded adversary A ,

$$\Pr \left[\begin{array}{l} s \neq s' \\ \wedge V^f(\text{pp}, z, s) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A^f(\text{pp}, z) \\ s' \leftarrow P^f(\text{pp}, z) \end{array} \right] \leq \zeta^*(\lambda, D).$$

Lemma 3.15 ([AY25, Theorem 6.3]). For every $\lambda, D \in \mathbb{N}$ with $\lambda \geq 1$ and $D \geq 2\lambda$ a power of two, there exists a proof-of-work PoW = (Setup, P, V) relative to a uniform puzzle generator Gen, with perfect correctness, $q_{\text{Setup}} = q_{\text{Gen}} = 0$, $q_P \leq D$, $q_V = O(\lambda \log D)$, and the following strong soundness guarantee. For every $q_{A_1}, q_{A_2} \in \mathbb{N}$ with $q_{A_2} \leq D/4$, and any adversary $A = (A_1, A_2)$ such that A_1 makes at most q_{A_1} queries and outputs an auxiliary string aux, and A_2 makes at most q_{A_2} online queries, we have

$$\Pr \left[V^f(\text{pp}, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \text{aux} \leftarrow A_1^f(\text{pp}) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A_2^f(\text{aux}, z) \end{array} \right] \leq \frac{q_{A_1} + q_{A_2} + 1}{2^\lambda}.$$

In particular, the error is negligible when q_{A_1} and q_{A_2} are polynomially bounded in λ .

Proof. Use the Merkle-tree PoW of [AY25, Construction 6.2]. The setup outputs $pp = (1^\lambda, D)$, and the puzzle generator samples $z \leftarrow \{0, 1\}^\lambda$. We take the Merkle root together with its opening proof as the solution s . The correctness and query bounds follow from the construction, and the strong soundness bound follows from [AY25, Theorem 6.3]. $\square$

Lemma 3.16. *For every $\lambda \in \mathbb{N}$ with $\lambda \geq 1$ and $D = 2^d$ for an integer $0 \leq d \leq \lambda$, there exists a proof-of-work PoW = (Setup, P, V) relative to a uniform puzzle generator Gen, with deterministic prover, difficulty parameter D , $q_{\text{Setup}} = q_{\text{Gen}} = 0$, $q_P = \lambda \cdot D$, $q_V = 1$, and correctness error $\alpha(\lambda, D, q_P) \leq e^{-\lambda}$. Moreover, for every $k \geq 1$ and every expected query bound $q_A \in [0, \infty)$, PoW has k -fold non-amortization with $\gamma(\lambda, D, q_A) = 0$ according to Definition 3.12.*

Proof. We construct the preimage PoW as follows. All tuple encodings below are injective.

- $\text{Setup}^f(1^\lambda, D)$:
 1. Set $q_P := \lambda \cdot D$, $\mathcal{Z} := \{0, 1\}^\lambda$, and $\mathcal{S} := [q_P] \cup \{\perp\}$.
 2. Output $pp := (1^\lambda, D)$, which determines these spaces.
- $\text{Gen}^f(pp)$:
 1. Sample $z \leftarrow \mathcal{Z}$ uniformly at random.
 2. Output z .
- $P^f(pp, z)$:
 1. For $s = 1, \dots, q_P$, query $f(pp, z, s)$. If its first d bits are zero, output s and halt.
 2. Output $\perp$.
- $V^f(pp, z, s)$:
 1. If $z \notin \mathcal{Z}$, $s \notin [q_P]$, output 0.
 2. Query $f(pp, z, s)$. Output 1 if its first d bits are zero, and 0 otherwise.

By construction, the setup and puzzle generator make no oracle queries, the prover makes at most q_P queries, and the verifier makes at most one query. The solution is generated deterministically.

Correctness. The honest prover tests distinct nonces in a fixed order until it finds an accepted nonce. Each trial succeeds independently with probability $2^{-d} = 1/D$. Thus, the probability that the verifier rejects an honestly generated solution is

$$\alpha(\lambda, D, q_P) = \left(1 - \frac{1}{D}\right)^{q_P} \leq e^{-\lambda}.$$

The expected number of queries made by the honest prover satisfies

$$\mathbb{E}[\text{honest prover queries}] = \sum_{i=0}^{q_P-1} \left(1 - \frac{1}{D}\right)^i = D \left(1 - \left(1 - \frac{1}{D}\right)^{q_P}\right) \leq D.$$

Non-amortization. For every $k \geq 1$ and $q_A \in [0, \infty)$, let A be an adversary in Definition 3.12 that makes at most q_A queries in expectation. Let Q denote its total number of queries and Q_{fresh} the number of distinct oracle addresses it queries in that experiment. Each distinct accepted puzzle–solution pair corresponds to a distinct queried address whose response begins with d zero bits, since the adversary verifies every output

pair. The event that the adversary makes its i -th fresh query is determined before it sees that response, which begins with d zero bits with probability $1/D$. Consequently,

$$\mathbb{E}[N] \leq \frac{1}{D} \sum_{i \geq 1} \Pr[Q_{\text{fresh}} \geq i] = \frac{\mathbb{E}[Q_{\text{fresh}}]}{D} \leq \frac{\mathbb{E}[Q]}{D} \leq \frac{q_A}{D}.$$

All queries used to verify outputs or obtain oracle-dependent information are included in Q . The argument applies to random stopping times and every number of outputs. Thus, PoW has k -fold non-amortization with $\gamma = 0$. $\square$

4 From non-amortization to computational uniqueness

We formally state and prove Reduction 1 of Section 2.1. Specifically, we show that every strong and non-amortizing PoW can be transformed into a strong and computationally unique PoW (Theorem 4.1). The transformation lowers the difficulty by a parameter h , preserves the verifier query complexity and the correctness error, transfers strong soundness at the cost of h online queries, and bounds the computational uniqueness error by the correctness and non-amortization errors of the original PoW.

We model the random oracle $f \in \mathcal{O}(\lambda)$ as a uniformly random function $f : \mathcal{X} \rightarrow [\mathcal{M}]$ for some finite address space $\mathcal{X}$ and response alphabet $[\mathcal{M}]$, where $\mathcal{M} \geq 1$ is an integer. We assume that the public parameter pp and puzzle z contain the complete query–answer transcripts of their generations (the generation algorithms can append these transcripts to their outputs). Then, conditioned on pp , z , all oracle answers outside these transcripts are independent and uniform. Throughout the proof, probabilities for fixed public parameters and puzzle are taken under this conditional distribution.

Theorem 4.1. *For every $\lambda, D, q_P \in \mathbb{N}$ with $\lambda, D, q_P \geq 1$, fixed integer $0 \leq h < D$ with $h \leq q_P$, let $\text{PoW} = (\text{Setup}, P, V)$ be a proof-of-work relative to some q_{Gen} -query puzzle generator Gen , that has setup query complexity q_{Setup} , verifier query complexity q_V , correctness error α , strong soundness error ϵ , and 2-fold non-amortization γ . Suppose that P halts almost surely, has cutoff q_P , and makes at most D queries in expectation. Then, there exists a proof-of-work system $\text{PoW}_{\text{alt}} = (\text{Setup}_{\text{alt}}, P_{\text{alt}}, V_{\text{alt}})$ with the following properties relative to a $(q_{\text{Gen}} + h)$ -query puzzle generator Gen_{alt} ,*

- *The difficulty parameter is $D_{\text{alt}} = D - h$. The setup makes at most q_{Setup} queries, the prover makes at most $q_P - h$ queries, and the verifier makes at most q_V queries.*
- *Correctness error $\alpha_{\text{alt}}(\lambda, D_{\text{alt}}, q_P - h) \leq \alpha(\lambda, D, q_P)$.*
- *Strong soundness error $\epsilon_{\text{alt}}(\lambda, D_{\text{alt}}, q_{A_1}, q_{A_2}) \leq \epsilon(\lambda, D, q_{A_1}, h + q_{A_2})$.*
- *Computational uniqueness error*

$$\zeta_{\text{alt}}(\lambda, D_{\text{alt}}, q_A) \leq \alpha(\lambda, D, q_P) + \gamma(\lambda, D, q_{\text{Gen}} + D + q_A + 2q_V) + \frac{q_{\text{Gen}} + q_A + 2q_V}{D}.$$

The soundness and uniqueness bounds hold for every $q_{A_1}, q_{A_2}, q_A \in \mathbb{N}$. The computational uniqueness bound also holds for adversaries making at most q_A queries in expectation, with no oracle-dependent input other than pp and z_{alt} .

Proof. Split P into two phases.

- $\text{st} \xleftarrow{\text{tr}} P_{1 \rightarrow h}^f(\text{pp}, z)$: On input the public parameter pp and puzzle z , runs for the first h queries and outputs its state, stopping earlier if the prover halts.
- $s \leftarrow P_{h \rightarrow q_P}^f(\text{st}; \text{pp}, z)$: Resume from st after the first h queries and output the solution. If the next query would exceed $q_P - h$ additional queries, output $\perp$. If the saved execution has already halted, return its output without making any queries.

We construct the new puzzle generator as follows.

$\text{Gen}_{\text{alt}}^f(\text{pp})$:

1. Run $z \leftarrow \text{Gen}^f(\text{pp})$.
2. Run $\text{st} \xleftarrow{\text{tr}} P_{1 \rightarrow h}^f(\text{pp}, z)$.
3. Output $z_{\text{alt}} := (z, \text{st}, \text{tr})$.

The constructed puzzle generator makes at most $q_{\text{Gen}} + h$ queries. All properties of PoW_{alt} are measured relative to Gen_{alt} .

Set $D_{\text{alt}} := D - h$. $\text{Setup}_{\text{alt}}(1^\lambda, D_{\text{alt}})$ runs $\text{Setup}(1^\lambda, D)$ and attaches a fixed description of the residual puzzle format and generator. For simplicity, we omit this fixed wrapper and continue to write pp for the public parameters of PoW_{alt} . We construct the prover P_{alt} and verifier V_{alt} as follows.

- $P_{\text{alt}}^f(\text{pp}, z_{\text{alt}} := (z, \text{st}, \text{tr}))$:
 1. Run $s \leftarrow P_{h \rightarrow q_P}^f(\text{st}; \text{pp}, z)$.
 2. Output s .
- $V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s)$:
 1. Parse z_{alt} as $(z, \text{st}, \text{tr})$. If parsing fails, output 0.
 2. Run $b \leftarrow V^f(\text{pp}, z, s)$.
 3. Output b .

By construction, P_{alt} makes at most $q_P - h$ queries to the random oracle.

Correctness. The output of P_{alt} is exactly the output of the honest prover with cutoff q_P . The correctness bound for the original PoW therefore gives

$$\begin{aligned}
 \alpha(\lambda, D, q_P) &\geq \Pr \left[V^f(\text{pp}, z, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow P^f(\text{pp}, z) \end{array} \right] \\
 &= \Pr \left[V^f(\text{pp}, z, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ \text{st} \xleftarrow{\text{tr}} P_{1 \rightarrow h}^f(\text{pp}, z) \\ s \leftarrow P_{h \rightarrow q_P}^f(\text{st}; \text{pp}, z) \end{array} \right] \\
 &= \Pr \left[V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow P_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right].
 \end{aligned}$$

Thus, $\alpha_{\text{alt}}(\lambda, D_{\text{alt}}, q_P - h) \leq \alpha(\lambda, D, q_P)$.

Strong soundness. Given an adversary $B = (B_1, B_2)$ that makes at most q_{A_1} preprocessing queries and q_{A_2} online queries for the strong soundness experiment against PoW_{alt} , we build an adversary $A = (A_1, A_2)$ for the strong soundness experiment against PoW.

- $A_1^f(\text{pp})$:
 1. Run $\text{aux}' \leftarrow B_1^f(\text{pp})$.
 2. Output $\text{aux} := (\text{aux}', \text{pp})$.
- $A_2^f(\text{aux} := (\text{aux}', \text{pp}), z)$:
 1. Run $\text{st} \xleftarrow{\text{tr}} P_{1 \rightarrow h}^f(\text{pp}, z)$.
 2. Set $z_{\text{alt}} := (z, \text{st}, \text{tr})$.

3. Run $s \leftarrow B_2^f(\text{aux}', z_{\text{alt}})$.
4. Output s .

A makes at most $h + q_{A_2}$ online queries. Its preprocessing stage makes no additional oracle queries and includes pp in the auxiliary string.

Strong soundness of PoW gives

$$\begin{aligned}
\epsilon(\lambda, D, q_{A_1}, h + q_{A_2}) &\geq \Pr \left[V^f(\text{pp}, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \text{aux} \leftarrow A_1^f(\text{pp}) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A_2^f(\text{aux}, z) \end{array} \right] \\
&= \Pr \left[V^f(\text{pp}, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ \text{aux} \leftarrow B_1^f(\text{pp}) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ \text{st} \xleftarrow{\text{tr}} P_{1 \rightarrow h}^f(\text{pp}, z) \\ s \leftarrow B_2^f(\text{aux}, (z, \text{st}, \text{tr})) \end{array} \right] \\
&= \Pr \left[V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ \text{aux} \leftarrow B_1^f(\text{pp}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow B_2^f(\text{aux}, z_{\text{alt}}) \end{array} \right].
\end{aligned}$$

Therefore, $\epsilon_{\text{alt}}(\lambda, D_{\text{alt}}, q_{A_1}, q_{A_2}) \leq \epsilon(\lambda, D, q_{A_1}, h + q_{A_2})$.

Computational uniqueness. Given a q_A -query adversary B for the computational uniqueness experiment against PoW_{alt} , we build an adversary A for the 2-fold non-amortization experiment against PoW.

$A^f(\text{pp})$:

1. Run $z_{\text{alt}} := (z, \text{st}, \text{tr}) \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp})$.
2. Run $s \leftarrow B^f(\text{pp}, z_{\text{alt}})$.
3. Run $s' \leftarrow P_{\text{alt}}^f(\text{pp}, z_{\text{alt}})$ from the saved state.
4. Run V on each of the two candidate pairs, making at most $2q_V$ additional queries.
5. Output $\{(z, s), (z, s')\}$.

The expected number of queries made by A is at most $q_{\text{Gen}} + D + q_A + 2q_V$.

Let N denote the number of distinct accepted puzzle–solution pairs, namely

$$N := \left| \left\{ (z^{(i)}, s^{(i)}) \mid \begin{array}{l} i \in [k], \\ V^f(\text{pp}, z^{(i)}, s^{(i)}) = 1 \end{array} \right\} \right|.$$

By the 2-fold non-amortization property of PoW, the expected number of distinct accepted pairs satisfies the following bound.

$$\frac{q_{\text{Gen}} + D + q_A + 2q_V}{D} + \gamma(\lambda, D, q_{\text{Gen}} + D + q_A + 2q_V) \geq \mathbb{E} \left[N \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \{(z, s), (z, s')\} \leftarrow A^f(\text{pp}) \end{array} \right]$$

$$= \mathbb{E} \left[N \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} := (z, \text{st}, \text{tr}) \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow \text{B}^f(\text{pp}, z_{\text{alt}}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right].$$

Let E_p denote the event that s' is accepted by the verifier, and let E_B denote the event that B wins the computational uniqueness game. Observe that if s' is accepted by the verifier and B does not win the computational uniqueness game, then $N \geq 1$; similarly, if s' is not accepted by the verifier but s is, then $N = 1$. If the output of the honest prover is accepted and B wins, then $N = 2$. Therefore, the expected value of N is bounded as follows.

$$\begin{aligned} \mathbb{E}[N] &\geq \Pr \left[\begin{array}{l} (s = s' \\ \vee V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) \neq 1) \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s') = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow \text{B}^f(\text{pp}, z_{\text{alt}}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right] \\ &\quad + \Pr \left[\begin{array}{l} s \neq s' \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) = 1 \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s') \neq 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow \text{B}^f(\text{pp}, z_{\text{alt}}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right] \\ &\quad + 2 \cdot \Pr \left[\begin{array}{l} s \neq s' \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) = 1 \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s') = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow \text{B}^f(\text{pp}, z_{\text{alt}}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right] \\ &\geq \Pr \left[V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s') = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right] \\ &\quad + \Pr \left[\begin{array}{l} s \neq s' \\ \wedge V_{\text{alt}}^f(\text{pp}, z_{\text{alt}}, s) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}_{\text{alt}}^f(1^\lambda, D_{\text{alt}}) \\ z_{\text{alt}} \leftarrow \text{Gen}_{\text{alt}}^f(\text{pp}) \\ s \leftarrow \text{B}^f(\text{pp}, z_{\text{alt}}) \\ s' \leftarrow \text{P}_{\text{alt}}^f(\text{pp}, z_{\text{alt}}) \end{array} \right] \\ &\geq 1 - \alpha(\lambda, D, q_P) + \zeta_{\text{alt}}(\lambda, D_{\text{alt}}, q_A). \end{aligned}$$

Re-organizing the terms gives

$$\zeta_{\text{alt}}(\lambda, D_{\text{alt}}, q_A) \leq \alpha(\lambda, D, q_P) + \gamma(\lambda, D, q_{\text{Gen}} + D + q_A + 2q_V) + \frac{q_{\text{Gen}} + q_A + 2q_V}{D}. \quad \square$$

5 From computational uniqueness to statistical uniqueness

We formally state and prove Reduction 2 of Section 2.1. Specifically, we show that restricting the verifier of a computationally unique PoW yields a statistically unique PoW (Theorem 5.3). The restriction preserves the difficulty, the query complexities, and strong soundness, and it increases the correctness error by at most $8q_v$ times the computational uniqueness error, which also bounds the statistical uniqueness error. As a corollary, we answer an open question raised in [GRY25] (Corollary 5.4).

Definition 5.1. For any $\lambda, D \in \mathbb{N}$, fixed public parameter pp and puzzle z , and any proposed output s , let τ be a complete accepting transcript of $V^f(\text{pp}, z, s)$ for some $f \in \mathcal{O}(\lambda)$, and let m be the number of oracle addresses not fixed by pp and z . Define the following:

- A query–answer transcript is a finite sequence $\text{tr} = ((q_i, a_i))_{i \in [n]}$ of oracle queries $q_i \in \mathcal{X}$ and answers $a_i \in [\mathcal{M}]$. An oracle $f \in \mathcal{O}(\lambda)$ is consistent with tr if $f(q_i) = a_i$ for every $i \in [n]$. Recall that pp and z contain the query–answer transcripts of their generation. We write $\mathcal{O}[\text{pp}, z]$ for the set of oracles consistent with both of these transcripts, and $\mathcal{O}[\text{pp}, z, \tau]$ for the set of oracles in $\mathcal{O}[\text{pp}, z]$ that are also consistent with τ .
- $\omega := (\text{pp}, z, s, \tau)$.
- $\mathcal{F}_\omega$ is the set of all oracles $f \in \mathcal{O}[\text{pp}, z, \tau]$ that satisfy $P^f(\text{pp}, z) = s$.
- The agreement rate is defined as

$$r_\omega := \Pr[P^f(\text{pp}, z) = s \mid f \in \mathcal{O}[\text{pp}, z, \tau]] = \frac{\Pr[f \in \mathcal{F}_\omega]}{\Pr[f \in \mathcal{O}[\text{pp}, z, \tau]]} = \frac{|\mathcal{F}_\omega|}{|\mathcal{O}[\text{pp}, z, \tau]|}.$$

We say (pp, z, s, τ) is saturated if $r_\omega \geq \frac{1}{2}$.

- For any $z \in \mathcal{Z}$, the certificate disagreement is

$$\Delta_{\text{pp}, z} := \sum_{\omega} \Pr[f \in \mathcal{F}_\omega] (1 - r_\omega) = \sum_{\omega} \frac{|\mathcal{F}_\omega|}{\mathcal{M}^m} (1 - r_\omega),$$

where the sums range over all $\omega = (\text{pp}, z, s, \tau)$ with the fixed pp and z . Let $\Delta := \mathbb{E}_{\text{pp}, z} [\Delta_{\text{pp}, z}]$ be the expected certificate disagreement over pp and z .

The agreement rate is the probability that an oracle consistent with the verifier transcript τ makes the honest prover output s . The certificate disagreement measures how often accepting verifier transcripts fail to pin down the prover’s output.

Construction 5.2. Let $\text{PoW} = (\text{Setup}, P, V)$ be a proof of work relative to a puzzle generator Gen . The **restricted system** $(\text{Setup}, P, \hat{V})$ keeps Setup , P , and Gen unchanged and replaces the verifier as follows.

$\hat{V}^f(\text{pp}, z, s)$:

1. Run $b \leftarrow V^f(\text{pp}, z, s)$ and let τ be the transcript of this execution.
2. Set $\omega := (\text{pp}, z, s, \tau)$. Output 1 if $b = 1$ and $r_\omega \geq 1/2$, and 0 otherwise.

We write $\hat{\alpha}$, $\hat{\zeta}$, $\hat{\zeta}^*$, and $\hat{\epsilon}$ for the correctness, computational uniqueness, statistical uniqueness, and strong soundness errors of the restricted system.

Theorem 5.3. For every $\lambda, q_{A_1}, q_{A_2} \in \mathbb{N}$ such that $q_v \geq 1$, let $\text{PoW} = (\text{Setup}, P, V)$ be a proof-of-work relative to a specified puzzle generator Gen with difficulty parameter D , setup query complexity q_{Setup} , prover query complexity q_p , verifier query complexity q_v , correctness error α , strong soundness error ϵ , and computational uniqueness error ζ . Then the restricted system $(\text{Setup}, P, \hat{V})$ of Construction 5.2 has the following properties:

- It has the same difficulty parameter D , setup query q_{Setup} , prover query q_P , and $\widehat{V}$ makes at most q_V queries.
- Correctness error $\widehat{\alpha}(\lambda, D, q_P) \leq \alpha(\lambda, D, q_P) + 8 \cdot q_V \cdot \zeta(\lambda, D, q_P)$.
- Strong soundness error $\widehat{\epsilon}(\lambda, D, q_{A_1}, q_{A_2}) \leq \epsilon(\lambda, D, q_{A_1}, q_{A_2})$.
- Statistical uniqueness error $\widehat{\zeta}^*(\lambda, D) \leq 8 \cdot q_V \cdot \zeta(\lambda, D, q_P)$.

The following corollary answers an open question raised in [GRY25] for search problems with negligible correctness and computational uniqueness errors and polynomial verifier query complexity.

Corollary 5.4. *Let PoW be as in Theorem 5.3, whose setup and puzzle generator output fixed values pp and z without making oracle queries. For every non-zero constant δ such that*

$$\alpha(\lambda, D, q_P) + 16 \cdot q_V \cdot \zeta(\lambda, D, q_P) \leq \delta \leq 10^{-3},$$

there exists an adversary F that makes at most q_V^2/δ queries and satisfies

$$\Pr \left[V^f(pp, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ s \leftarrow F^f(pp, z) \end{array} \right] \geq 1 - 6\sqrt{\delta}.$$

The conclusion holds for every finite response alphabet.

Proof. Apply Theorem 5.3 with no setup or puzzle-generation queries. Its proof uses only the finite product distribution of oracle answers, and applies to every finite response alphabet. The probability that the restricted verifier does not have exactly one accepted output is at most

$$\widehat{\alpha}(\lambda, D, q_P) + \widehat{\zeta}^*(\lambda, D) \leq \alpha(\lambda, D, q_P) + 16 \cdot q_V \cdot \zeta(\lambda, D, q_P) \leq \delta.$$

By [GRY25, Theorem 4.2], there exists an adversary making at most q_V^2/δ queries that finds an output accepted by the restricted verifier with probability at least $1 - 6\sqrt{\delta}$. Every such output is also accepted by the original verifier, which proves the claim. $\square$

Section 5.1 proves technical lemmas, and Section 5.2 proves Theorem 5.3.

5.1 Technical lemmas

Lemma 5.5. *For every real $x \geq 1$, $\frac{(x+1)^{(x+1)}}{x^x} \leq 4x$.*

Proof. We start by taking the logarithm as follows.

$$\log \left(\frac{(x+1)^{(x+1)}}{4x \cdot x^x} \right) = (x+1) \log(x+1) - (x+1) \log x - \log 4 = (x+1) \log \left(1 + \frac{1}{x} \right) - \log 4.$$

The derivative of the expression above is $\log \left(1 + \frac{1}{x} \right) - \frac{1}{x}$. This derivative is negative for every $x \geq 1$, since $\log(1+y) < y$ for every positive y . This means the function above is decreasing over $x \geq 1$. At $x = 1$, we get,

$$2 \log 2 - \log 4 = 0,$$

which implies the logarithm value is smaller than 0 for every $x \geq 1$, and, therefore, $\frac{(x+1)^{(x+1)}}{x^x} \leq 4x$. $\square$

Lemma 5.6. *For every $\lambda, D, q_V \in \mathbb{N}$ such that $q_V \geq 1$, $\theta \in (0, 1)$, the following holds.*

$$\Delta \leq 4 \cdot q_V \cdot \zeta(\lambda, D, q_P).$$

Proof. For $f \in \mathcal{O}(\lambda)$ and $\theta \in (0, 1)$, define the θ -resampling procedure Resample_θ as follows. For any two oracles f, g , if g is a θ -resampling of f , then f is also a θ -resampling of g . Given an oracle function f and a rate $\theta \in (0, 1)$, the θ -resampling procedure for f is as follows.

$\text{Resample}_\theta(f, \text{pp}, z)$:

1. For every query q constrained by pp, z , set $g(q) = f(q)$.
2. For every query q in the domain of f and not constrained by pp, z , do the following.
 - (a) Sample $c \leftarrow \text{Bernoulli}(\theta)$.
 - (b) If $c = 0$, set $g(q) = f(q)$; otherwise if $c = 1$, sample $g(q) \leftarrow [\mathcal{M}]$ uniformly.
3. Output the new oracle function g .

We construct the following adversary B_θ in the computational uniqueness experiment for PoW. Given a rate $\theta \in (0, 1)$, an auxiliary state pp and a puzzle $z \in \mathcal{Z}$, the adversary B_θ is as follows.

$B_\theta^f(\text{pp}, z)$:

1. Run the honest prover $s \leftarrow P^g(\text{pp}, z)$, simulating g as follows. Maintain a local cache of query-answer pairs. For every query q made by P , if it is constrained by transcripts contained in pp, z or the local cache, respond with the answer from that transcript; otherwise, do the following.
 - (a) Sample $c \leftarrow \text{Bernoulli}(\theta)$.
 - (b) If $c = 0$, answer the query with $f(q)$; otherwise if $c = 1$, sample the answer uniformly from $[\mathcal{M}]$.
 - (c) Store this query-answer pair into the local cache.
2. Output s .

By construction, B_θ has no preprocessing stage, makes at most q_P online queries, and simulates Resample_θ . By the computational uniqueness of PoW, we have

$$\begin{aligned} \zeta(\lambda, D, q_P) &\geq \Pr \left[\begin{array}{c|c} s \neq s' \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow B^f(\text{pp}, z) \\ s' \leftarrow P^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} V^f(\text{pp}, z, s) \end{array} \right] \\ &= \Pr \left[\begin{array}{c|c} s \neq s' \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ g \leftarrow \text{Resample}_\theta(f, \text{pp}, z) \\ s \leftarrow P^g(\text{pp}, z) \\ s' \leftarrow P^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} V^f(\text{pp}, z, s) \end{array} \right]. \end{aligned}$$

On the event that Resample_θ does not change any queries in τ , both f and g must be consistent with τ . Then, $s \leftarrow P^g(\text{pp}, z)$ implies $g \in \mathcal{F}_\omega$, and $s \neq s'$ implies $f \notin \mathcal{F}_\omega$.

For any possible output s and verifier accepting transcript τ , define E_ω to be the event that $f \notin \mathcal{F}_\omega$, none of the query responses in τ was changed during θ -resampling process of B_θ , and $g \in \mathcal{F}_\omega$, i.e.

$$E_\omega := [g \in \mathcal{F}_\omega] \cap [\text{Resample}_\theta \text{ does not change } \tau] \cap [f \notin \mathcal{F}_\omega].$$

On E_ω for any fixed $\omega = (\text{pp}, z, s, \tau)$, the adversary B_θ outputs s since $g \in \mathcal{F}_\omega$. Moreover, since θ -resampling changes none of the query responses in τ , f and g must agree on all query locations in τ . This means, $f \notin \mathcal{F}_\omega$

implies $s^* \leftarrow P^f(\text{pp}, z)$ and $s^* \neq s$. Therefore, if E_ω occurs for the fixed pair of output and verifier transcript and the verifier accepts the honestly generated output, then B_θ wins the computational uniqueness game.

We prove that the events E_ω are disjoint for different ω . For contradiction, assume the sets $\mathcal{F}_\omega$ are not pairwise disjoint. Then, for some $(s, \tau) \neq (s', \tau')$, there must exist some oracle g that belongs to both $\mathcal{F}_\omega$ and $\mathcal{F}_{\omega'}$, where $\omega' = (\text{pp}, z, s', \tau')$. s and s' must be the same since $s = s' = P^g(\text{pp}, z)$. Moreover, determinism of the verifier implies $\tau = \tau'$. This is a contradiction. Therefore, the sets $\mathcal{F}_\omega$ must be disjoint for different ω , and hence the events E_ω are disjoint. Summing their probabilities gives a lower bound on the success probability of B_θ . Consequently,

$$\zeta(\lambda, D, q_P) \geq \sum_{\omega} \Pr \left[E_\omega \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ g \leftarrow \text{Resample}_\theta(f, \text{pp}, z) \end{array} \right].$$

We now bound $\Pr[E_\omega]$ from below for fixed (pp, z, s, τ) . Since the resampled queries are independent of the function g , we have

$$\Pr[g \in \mathcal{F}_\omega \wedge \tau \text{ unchanged}] = \Pr[g \in \mathcal{F}_\omega] \cdot \Pr[\tau \text{ unchanged}] \geq \Pr[g \in \mathcal{F}_\omega] (1 - \theta)^{q_V}.$$

We next bound the conditional probability of $f \notin \mathcal{F}_\omega$ given that $g \in \mathcal{F}_\omega$ and τ is unchanged.

$$\begin{aligned} & \Pr[f \notin \mathcal{F}_\omega \mid g \in \mathcal{F}_\omega \wedge \tau \text{ unchanged}] \\ &= 1 - \Pr[f \in \mathcal{F}_\omega \mid g \in \mathcal{F}_\omega \wedge \tau \text{ unchanged}] \\ &= 1 - \frac{\Pr[f \in \mathcal{F}_\omega \wedge g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}]}{\Pr[g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}]} \\ &= 1 - \frac{\Pr[f \in \mathcal{F}_\omega \wedge g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}]}{\Pr[g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau]]} \\ &= 1 - \frac{\Pr[f \in \mathcal{F}_\omega \wedge g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}]}{r_\omega}. \end{aligned} \tag{2}$$

The third equality uses the independence of g and the event that τ is unchanged. We need only consider ω with $r_\omega > 0$, since $\mathcal{F}_\omega$ has probability zero when $r_\omega = 0$.

To compute the conditional probability in the numerator, let X be 1 when $g \in \mathcal{F}_\omega$ given that g is consistent with τ and τ is unchanged, and 0 otherwise; similarly, define Y for f . Then the following hold.

- Since X is an indicator, $\mathbb{E}[X^2] = \mathbb{E}[X]$, and since g is independent of the event that τ is unchanged,

$$\mathbb{E}[X] = \Pr[g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}] = \Pr[g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau]] = r_\omega.$$

- By the symmetry of the joint distribution of f and g ,

$$\mathbb{E}[Y] = \Pr[f \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}] = \Pr[f \in \mathcal{F}_\omega \mid f \in \mathcal{O}[\text{pp}, z, \tau]] = r_\omega.$$

- The numerator in Eq. (2) is

$$\Pr[f \in \mathcal{F}_\omega \wedge g \in \mathcal{F}_\omega \mid g \in \mathcal{O}[\text{pp}, z, \tau] \wedge \tau \text{ unchanged}] = \mathbb{E}[XY].$$

We prove by induction that $\mathbb{E}[XY] \leq (1 - \theta)\mathbb{E}[X^2] + \theta r_\omega^2$. For the induction, we consider the more general case when X, Y evaluate the same $[0, 1]$ -valued function on g, f , respectively, with common expectation r_ω .

- Base case: there is only one oracle position.
 - With probability θ , we resample the answer uniformly at random. Then X and Y are independent, in which case $\mathbb{E}[XY] = \mathbb{E}[X] \mathbb{E}[Y] = r_\omega^2$.
 - With probability $1 - \theta$, we do not resample this position. Then $Y = X$, and $\mathbb{E}[XY] = \mathbb{E}[X^2]$. Thus, in the base case, $\mathbb{E}[XY] = (1 - \theta)\mathbb{E}[X^2] + \theta r_\omega^2$.
- Inductive case: For m oracle positions, assume that the inequality holds for any number of positions fewer than m . Define $\bar{X}, \bar{Y}$ by averaging X, Y over the last position while keeping all the other positions fixed, respectively. Observe that $\mathbb{E}[X - \bar{X}] = \mathbb{E}[Y - \bar{Y}] = 0$ by definition. Expanding the product gives

$$\begin{aligned}\mathbb{E}[XY] &= \mathbb{E}[\bar{X}\bar{Y} + \bar{X}(Y - \bar{Y}) + \bar{Y}(X - \bar{X}) + (X - \bar{X})(Y - \bar{Y})] \\ &= \mathbb{E}[\bar{X}\bar{Y}] + \mathbb{E}[(X - \bar{X})(Y - \bar{Y})].\end{aligned}$$

By the induction hypothesis, $\mathbb{E}[\bar{X}\bar{Y}] \leq (1 - \theta)\mathbb{E}[\bar{X}^2] + \theta r_\omega^2$. To bound the second term, consider the resampling decision at the last position.

- With probability θ , we resample the answer at the last position uniformly. Then, conditional on both vectors of remaining coordinates, $(X - \bar{X})$ and $(Y - \bar{Y})$ are independent, and $\mathbb{E}[(X - \bar{X})(Y - \bar{Y})] = \mathbb{E}[X - \bar{X}] \mathbb{E}[Y - \bar{Y}] = 0$.
- With probability $(1 - \theta)$, we do not resample this position. Then, using $ab \leq \frac{a^2 + b^2}{2}$, we get

$$\mathbb{E}[(X - \bar{X})(Y - \bar{Y})] \leq \frac{\mathbb{E}[(X - \bar{X})^2 + (Y - \bar{Y})^2]}{2} = \mathbb{E}[(X - \bar{X})^2].$$

The second moment decomposes as

$$\mathbb{E}[X^2] = \mathbb{E}[(\bar{X} + (X - \bar{X}))^2] = \mathbb{E}[\bar{X}^2] + \mathbb{E}[(X - \bar{X})^2].$$

Combining these bounds gives

$$\mathbb{E}[XY] \leq (1 - \theta)\mathbb{E}[\bar{X}^2] + \theta r_\omega^2 + (1 - \theta)\mathbb{E}[(X - \bar{X})^2] = (1 - \theta)\mathbb{E}[X^2] + \theta r_\omega^2.$$

Substituting $\mathbb{E}[X^2] = r_\omega$ into the inequality proven by induction gives

$$\mathbb{E}[XY] \leq (1 - \theta)r_\omega + \theta r_\omega^2. \quad (3)$$

Substituting Eq. (3) into Eq. (2) and summing over ω gives

$$\begin{aligned}\sum_{\omega} \Pr[\mathbf{E}_\omega] &= \sum_{\omega} \Pr[f \notin \mathcal{F}_\omega \mid g \in \mathcal{F}_\omega \wedge \tau \text{ unchanged}] \\ &\quad \cdot \Pr[g \in \mathcal{F}_\omega \wedge \tau \text{ unchanged}] \\ &\geq \sum_{\omega} \left(1 - \frac{(1 - \theta)r_\omega + \theta r_\omega^2}{r_\omega}\right) \cdot \Pr[g \in \mathcal{F}_\omega] (1 - \theta)^{\text{qv}} \\ &= \sum_{\omega} \theta(1 - r_\omega) \cdot \Pr[g \in \mathcal{F}_\omega] (1 - \theta)^{\text{qv}} \\ &= \theta(1 - \theta)^{\text{qv}} \Delta_{\text{pp}, z}.\end{aligned}$$

Averaging over the public view (pp, z) , we have

$$\zeta(\lambda, \mathbf{D}, \mathbf{q}_\text{p}) \geq \theta(1 - \theta)^{\text{qv}} \Delta.$$

Differentiating $\theta(1 - \theta)^{q_v}$ gives

$$\frac{d}{d\theta}\theta(1 - \theta)^{q_v} = (1 - \theta)^{q_v} + \theta \frac{d}{d\theta}(1 - \theta)^{q_v} = (1 - \theta)^{q_v} - \theta q_v (1 - \theta)^{q_v-1} = (1 - \theta)^{q_v-1} (1 - (q_v + 1)\theta).$$

The derivative changes from positive to negative at $\theta = 1/(q_v + 1)$, where the expression attains its maximum.

Substituting $\theta = \frac{1}{q_v+1}$ into the original lower bound, we have

$$\zeta(\lambda, D, q_p) \geq \theta(1 - \theta)^{q_v} \Delta = \frac{1}{q_v + 1} \left(1 - \frac{1}{q_v + 1}\right)^{q_v} \Delta.$$

By Lemma 5.5, we conclude

$$\Delta \leq \frac{(q_v + 1)^{q_v+1}}{(q_v)^{q_v}} \cdot \zeta(\lambda, D, q_p) \leq 4 \cdot q_v \cdot \zeta(\lambda, D, q_p).$$

□

5.2 Proof of Theorem 5.3

The restricted verifier $\hat{V}$ makes the same queries as V : the agreement rate r_ω can be computed by exhaustive enumeration or supplied as nonuniform advice. Every output accepted by $\hat{V}$ is accepted by V , so strong soundness is preserved. It remains to bound correctness and statistical uniqueness.

Correctness Error. Suppose the original verifier accepts the prover's output $s \leftarrow P^f(pp, z)$, with verifier transcript τ . Then $f \in \mathcal{F}_\omega$. If $r_\omega \geq 1/2$, $\hat{V}$ also accepts s .

$$\begin{aligned} & \Pr \left[\hat{V}^f(pp, z, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ pp \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(pp) \\ s \leftarrow P^f(pp, z) \end{array} \right] \\ &= \Pr \left[\begin{array}{l} b \neq 1 \\ \vee (b = 1 \wedge r_\omega < 1/2) \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ pp \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(pp) \\ s \leftarrow P^f(pp, z) \\ b \xleftarrow{\tau} V^f(pp, z, s) \end{array} \right] \\ &\leq \Pr \left[V^f(pp, z, s) \neq 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ pp \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(pp) \\ s \leftarrow P^f(pp, z) \end{array} \right] \\ &\quad + \Pr \left[b = 1 \wedge r_\omega < 1/2 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ pp \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(pp) \\ s \leftarrow P^f(pp, z) \\ b \xleftarrow{\tau} V^f(pp, z, s) \end{array} \right] \end{aligned}$$

$$\leq \alpha(\lambda, D, q_P) + \Pr \left[b = 1 \wedge r_\omega < 1/2 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow \text{P}^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} \text{V}^f(\text{pp}, z, s) \end{array} \right].$$

Fix (pp, z) and write $\omega = (\text{pp}, z, s, \tau)$. The second probability is bounded as follows.

$$\begin{aligned} & \Pr \left[b = 1 \wedge r_\omega < 1/2 \mid \begin{array}{l} f \leftarrow \mathcal{O}[\text{pp}, z] \\ s \leftarrow \text{P}^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} \text{V}^f(\text{pp}, z, s) \end{array} \right] \\ &= \Pr \left[\begin{array}{l} f \in \mathcal{O}[\text{pp}, z, \tau] \\ \wedge r_\omega < 1/2 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}[\text{pp}, z] \\ s \leftarrow \text{P}^f(\text{pp}, z) \\ 1 \stackrel{\tau}{\leftarrow} \text{V}^f(\text{pp}, z, s) \end{array} \right] \\ &= \sum_{(s, \tau): r_\omega < 1/2} \Pr \left[\begin{array}{l} f \in \mathcal{O}[\text{pp}, z, \tau] \\ \wedge s = \text{P}^f(\text{pp}, z) \end{array} \mid f \leftarrow \mathcal{O}[\text{pp}, z] \right] \\ &= \sum_{(s, \tau): r_\omega < 1/2} \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \\ &\leq 2 \sum_{(s, \tau): r_\omega < 1/2} (1 - r_\omega) \cdot \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \\ &\leq 2\Delta_{\text{pp}, z}. \end{aligned}$$

Averaging over (pp, z) gives

$$\begin{aligned} & \Pr \left[b = 1 \wedge r_\omega < 1/2 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow \text{P}^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} \text{V}^f(\text{pp}, z, s) \end{array} \right] \\ &\leq 2 \cdot \mathbb{E}_{\text{pp}, z} [\Delta_{\text{pp}, z}] = 2\Delta. \end{aligned}$$

By Lemma 5.6, we conclude that

$$\hat{\alpha}(\lambda, D, q_P) \leq \alpha(\lambda, D, q_P) + 8 \cdot q_V \cdot \zeta(\lambda, D, q_P).$$

Statistical Uniqueness. Fix (pp, z) . Expanding the statistical uniqueness experiment gives

$$\begin{aligned} & \Pr \left[\begin{array}{l} s \neq s' \\ \wedge \hat{\text{V}}^f(\text{pp}, z, s) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}[\text{pp}, z] \\ s \leftarrow \text{A}^f(\text{pp}, z) \\ s' \leftarrow \text{P}^f(\text{pp}, z) \end{array} \right] \\ &= \Pr \left[\begin{array}{l} s \neq s' \\ \wedge b = 1 \\ \wedge r_\omega \geq 1/2 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}[\text{pp}, z] \\ s \leftarrow \text{A}^f(\text{pp}, z) \\ s' \leftarrow \text{P}^f(\text{pp}, z) \\ b \stackrel{\tau}{\leftarrow} \text{V}^f(\text{pp}, z, s) \end{array} \right] \end{aligned}$$

$$\begin{aligned}
&= \Pr \left[\begin{array}{c} s \neq P^f(\text{pp}, z) \\ \wedge f \in \mathcal{O}[\text{pp}, z, \tau] \\ \wedge r_\omega \geq 1/2 \end{array} \middle| \begin{array}{c} f \leftarrow \mathcal{O}[\text{pp}, z] \\ s \leftarrow A^f(\text{pp}, z) \\ 1 \stackrel{\tau}{\leftarrow} V^f(\text{pp}, z, s) \end{array} \right] \\
&\leq \sum_{(s, \tau): r_\omega \geq 1/2} \Pr \left[\begin{array}{c} f \in \mathcal{O}[\text{pp}, z, \tau] \\ \wedge s \neq P^f(\text{pp}, z) \end{array} \middle| f \leftarrow \mathcal{O}[\text{pp}, z] \right].
\end{aligned}$$

By the definition of $\mathcal{F}_\omega$, this sum is at most

$$\begin{aligned}
&\sum_{(s, \tau): r_\omega \geq 1/2} \left(\Pr [f \in \mathcal{O}[\text{pp}, z, \tau] \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \right. \\
&\quad \left. - \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \right) \\
&= \sum_{(s, \tau): r_\omega \geq 1/2} \left(\frac{\Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]]}{r_\omega} \right. \\
&\quad \left. - \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \right) \\
&= \sum_{(s, \tau): r_\omega \geq 1/2} \left(\frac{1 - r_\omega}{r_\omega} \right) \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \\
&\leq 2 \sum_{(s, \tau): r_\omega \geq 1/2} (1 - r_\omega) \cdot \Pr [f \in \mathcal{F}_\omega \mid f \leftarrow \mathcal{O}[\text{pp}, z]] \\
&\leq 2 \cdot \Delta_{\text{pp}, z}.
\end{aligned}$$

Averaging over (pp, z) gives

$$\begin{aligned}
&\Pr \left[\begin{array}{c} s \neq s' \\ \wedge \widehat{V}^f(\text{pp}, z, s) = 1 \end{array} \middle| \begin{array}{c} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A^f(\text{pp}, z) \\ s' \leftarrow P^f(\text{pp}, z) \end{array} \right] \\
&\leq 2 \cdot \mathbb{E}_{\text{pp}, z} [\Delta_{\text{pp}, z}] = 2\Delta.
\end{aligned}$$

Thus, by Lemma 5.6, we conclude,

$$\widehat{\zeta}^*(\lambda, D) \leq 2\Delta \leq 8 \cdot q_v \cdot \zeta(\lambda, D, q_p).$$

6 Impossibility of PoW

We chain the two reductions and apply the search bound of [GRY25] to prove Theorem 6.1, the formal version of Theorem 1. Specifically, we show that for every non-amortizing PoW with efficient verification, an adversary that makes no preprocessing queries and at most $D - \Omega(D/q_v)$ online queries solves a random puzzle with probability $\Omega(1/q_v)$, so the PoW is not strong.

We use the oracle model and the public generation transcripts of Section 5.

Theorem 6.1. *Let $\text{PoW} = (\text{Setup}, P, V)$ be a proof-of-work relative to a q_{Gen} -query puzzle generator Gen , with difficulty parameter $D \geq 1$, setup query complexity q_{Setup} , verifier query complexity $q_v \geq 1$, correctness error α , and 2-fold non-amortization error γ . Suppose that P halts almost surely, has cutoff q_p , and makes at most D queries in expectation. Fix a constant $\delta \in (0, 10^{-3}]$, and define $L := \delta \cdot D / (100 \cdot q_v)$. Suppose that $q_{\text{Gen}} + q_{\text{Setup}} + 2q_v + q_v^2/\delta \leq L$ and, for every expected query bound $0 \leq q_A \leq q_{\text{Gen}} + 2D + 2q_v$,*

$$\alpha(\lambda, D, q_p) + \gamma(\lambda, D, q_A) \leq \frac{\delta}{100 \cdot q_v}.$$

Then there exists an adversary $A = (A_1, A_2)$ such that A_1 makes no oracle queries and outputs an auxiliary string aux , and A_2 makes at most $q_{A_2} := D - L$ online queries and satisfies

$$\Pr \left[V^f(\text{pp}, z, s) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{pp} \leftarrow \text{Setup}^f(1^\lambda, D) \\ \text{aux} \leftarrow A_1^f(\text{pp}) \\ z \leftarrow \text{Gen}^f(\text{pp}) \\ s \leftarrow A_2^f(\text{aux}, z) \end{array} \right] > \frac{\delta}{100 \cdot q_v}.$$

Remark 6.2. For a fixed $\delta \in (0, 10^{-3}]$, the hypotheses of Theorem 6.1 hold for all sufficiently large λ whenever

$$q_v^3 = o(D), q_{\text{Setup}} + q_{\text{Gen}} = o(D/q_v), \alpha(\lambda, D, q_p) + \gamma(\lambda, D, q_A) = o(1/q_v),$$

where the last bound holds uniformly over $0 \leq q_A \leq q_{\text{Gen}} + 2D + 2q_v$. In particular, negligible correctness and non-amortization errors suffice when q_v is polynomially bounded in λ . The theorem then gives an adversary that makes at most $D - \Omega(D/q_v)$ online queries and succeeds with probability $\Omega(1/q_v)$. For $q_v = O(1)$, this is a constant lower bound on the strong soundness error at a constant-fraction online budget with no preprocessing queries; for growing q_v , the guaranteed saving tends to zero. The argument uses only the expected query bound of the honest prover.

As a sanity check, the preimage PoW of Lemma 3.16 satisfies the hypotheses with $q_v = 1$, $q_{\text{Setup}} = q_{\text{Gen}} = 0$, and $\gamma = 0$, so the theorem yields an adversary that saves a constant fraction of the queries with constant success probability. This matches the direct computation: the honest prover finds an accepted solution within $D/2$ queries with probability $1 - (1 - 1/D)^{D/2} \rightarrow 1 - e^{-1/2}$.

Section 6.1 chains the two reductions, and Section 6.2 proves Theorem 6.1.

6.1 Chaining the two reductions together

Corollary 6.3. *For every $\lambda \in \mathbb{N}$, let $\text{PoW} = (\text{Setup}, P, V)$ be a proof-of-work relative to a q_{Gen} -query puzzle generator Gen , with difficulty parameter $D \geq 1$, setup query complexity q_{Setup} , prover cutoff $q_p \geq 1$, verifier query complexity $q_v \geq 1$, correctness error α , strong soundness error ϵ , and 2-fold non-amortization error γ .*

Suppose that P halts almost surely, and makes at most D queries in expectation. Let $h \in [0, D)$ be an integer such that $h \leq q_P$. Then, there exists a proof-of-work system $\text{PoW}' = (\text{Setup}', P', V')$ with the following properties relative to a $(q_{\text{Gen}} + h)$ -query puzzle generator Gen_{alt} :

- The difficulty parameter is $D' := D - h$. Setup makes at most q_{Setup} queries, the prover makes at most $q_P - h$ queries, and the verifier makes at most q_V queries.
- Correctness error

$$\begin{aligned} & \alpha'(\lambda, D', q_P - h) \\ & \leq (1 + 8 \cdot q_V) \cdot \alpha(\lambda, D, q_P) \\ & \quad + 8 \cdot q_V \cdot \gamma(\lambda, D, q_{\text{Gen}} + D + q_P - h + 2q_V) \\ & \quad + 8 \cdot q_V \cdot \frac{q_{\text{Gen}} + q_P - h + 2q_V}{D}. \end{aligned}$$

- For every $q_{A_1}, q_{A_2} \in \mathbb{N}$, strong soundness error

$$\epsilon'(\lambda, D', q_{A_1}, q_{A_2}) \leq \epsilon(\lambda, D, q_{A_1}, h + q_{A_2}).$$

- Statistical uniqueness error

$$\zeta^{*'}(\lambda, D') \leq 8 \cdot q_V \cdot \alpha(\lambda, D, q_P) + 8 \cdot q_V \cdot \gamma(\lambda, D, q_{\text{Gen}} + D + q_P - h + 2q_V) + 8 \cdot q_V \cdot \frac{q_{\text{Gen}} + q_P - h + 2q_V}{D}.$$

Moreover, if the residual prover makes at most r queries in expectation over the honestly generated public view and oracle, the correctness and statistical uniqueness bounds remain valid with $q_P - h$ replaced by r in the argument of γ and in the numerator of the fractional term. The correctness errors are still evaluated at the displayed cutoffs.

Proof. Apply Theorem 4.1 with prefix length h to obtain PoW_{alt} , then apply Theorem 5.3 to obtain its restricted system PoW' . The query bounds follow immediately, and $D_{\text{alt}} = D' = D - h$. The two reductions give the following chains, evaluating computational uniqueness at query bound $q_P - h$.

Correctness.

$$\begin{aligned} & \alpha'(\lambda, D', q_P - h) \\ & \leq \alpha_{\text{alt}}(\lambda, D', q_P - h) + 8 \cdot q_V \cdot \zeta_{\text{alt}}(\lambda, D', q_P - h) \\ & \leq (1 + 8 \cdot q_V) \cdot \alpha(\lambda, D, q_P) \\ & \quad + 8 \cdot q_V \cdot \gamma(\lambda, D, q_{\text{Gen}} + D + q_P - h + 2q_V) \\ & \quad + 8 \cdot q_V \cdot \frac{q_{\text{Gen}} + q_P - h + 2q_V}{D}. \end{aligned}$$

Strong soundness.

$$\epsilon'(\lambda, D', q_{A_1}, q_{A_2}) \leq \epsilon_{\text{alt}}(\lambda, D', q_{A_1}, q_{A_2}) \leq \epsilon(\lambda, D, q_{A_1}, h + q_{A_2}).$$

Statistical uniqueness.

$$\zeta^{*'}(\lambda, D') \leq 8 \cdot q_V \cdot \zeta_{\text{alt}}(\lambda, D', q_P - h)$$

$$\begin{aligned}
&\leq 8 \cdot q_v \cdot \alpha(\lambda, D, q_p) \\
&\quad + 8 \cdot q_v \cdot \gamma(\lambda, D, q_{\text{Gen}} + D + q_p - h + 2q_v) \\
&\quad + 8 \cdot q_v \cdot \frac{q_{\text{Gen}} + q_p - h + 2q_v}{D}.
\end{aligned}$$

For the expected-query variant, the resampling adversary in the proof of Lemma 5.6, with $\theta = 1/(q_v + 1)$, makes at most r queries in expectation: conditioned on the public transcripts, its simulated oracle has the original product distribution, and each simulated prover query requires at most one real query. Apply the expected-query conclusion of Theorem 4.1 to this adversary. The proof of Theorem 5.3 uses precisely its success probability, so the same calculation gives the claimed replacements. $\square$

6.2 Proof of Theorem 6.1

Lemma 6.4 (Adapted from Theorem 4.1 in [GRY25]). *For every $\lambda \in \mathbb{N}$, let $\text{PoW} = (\text{Setup}, P, V)$ be a proof-of-work relative to a q_{Gen} -query puzzle generator Gen , with difficulty parameter D , setup query complexity q_{Setup} , prover query complexity q_p , verifier query complexity $q_v \geq 1$, correctness error α , strong soundness error ϵ , and statistical uniqueness error ζ^* . Then, for every constant δ such that $\alpha(\lambda, D, q_p) + \zeta^*(\lambda, D) \leq \delta \leq 10^{-3}$, and every $q_{A_2} \geq q_{\text{Setup}} + q_v^2/\delta$, $\epsilon(\lambda, D, 0, q_{A_2}) \geq 1 - 6\sqrt{\delta}$.*

Consider an honest prover P with query cutoff $D - L$. By the correctness of PoW , the probability that the output of this honest prover is accepted is at least $1 - \alpha(\lambda, D, D - L)$; let β denote this probability. If $\beta > \delta/(100 \cdot q_v)$, then running this honest prover proves the theorem. Otherwise, if $\beta \leq \delta/(100 \cdot q_v)$, then it must hold that $q_p > D - L$. This is because $q_p \leq D - L$ leads to the contradiction that $\beta \geq 1 - \alpha(\lambda, D, q_p) \geq 1 - \frac{\delta}{100q_v} > \frac{\delta}{100q_v}$. We continue to prove that the theorem statement holds in this case as well.

Set $h := D - 2L$, and let R be the number of honest prover queries remaining after its first h queries, which is zero if it halts earlier. The parameter bounds give $0 < L \leq D/10^5$, so $0 \leq h < D - L < q_p$. An honest execution that halts before h queries either succeeds within $D - L$ queries or contributes to the correctness error. Hence

$$\begin{aligned}
\mathbb{E}[R] &\leq D - h \cdot \Pr[\text{the honest prover makes at least } h \text{ queries}] \\
&\leq D - h + h \cdot \Pr[\text{the honest prover makes fewer than } h \text{ queries}] \\
&\leq 2L + D(\beta + \alpha(\lambda, D, q_p)) \\
&\leq 2L + \frac{\delta \cdot D}{50 \cdot q_v}.
\end{aligned}$$

Apply Corollary 6.3 with prefix length h and expected residual query bound $r = \mathbb{E}[R]$ to obtain PoW' , with $D' = D - h$ and residual cutoff $q_p - h$. Since $\mathbb{E}[R] \leq D$, the non-amortization argument below lies in the range of the error hypothesis. The corollary gives

$$\begin{aligned}
&\alpha'(\lambda, D', q_p - h) + \zeta^{*'}(\lambda, D') \\
&\leq (1 + 16 \cdot q_v) \cdot \alpha(\lambda, D, q_p) \\
&\quad + 16 \cdot q_v \cdot \gamma(\lambda, D, q_{\text{Gen}} + D + \mathbb{E}[R] + 2q_v) \\
&\quad + 16 \cdot q_v \cdot \frac{q_{\text{Gen}} + \mathbb{E}[R] + 2q_v}{D} \\
&\leq \frac{\delta}{100 \cdot q_v} + 16 \cdot q_v \left(\frac{\delta}{100 \cdot q_v} + \frac{q_{\text{Gen}} + 2L + 2q_v}{D} + \frac{\delta}{50 \cdot q_v} \right)
\end{aligned}$$

$$\begin{aligned}
&\leq \frac{\delta}{100 \cdot q_v} + 16 \cdot q_v \left(\frac{\delta}{100 \cdot q_v} + \frac{3\delta}{100 \cdot q_v} + \frac{\delta}{50 \cdot q_v} \right) \\
&\leq \frac{97\delta}{100}.
\end{aligned}$$

Here we used the bounds on $\mathbb{E}[R]$ and β , together with the query hypothesis, which gives $\frac{q_{\text{Gen}} + 2q_v}{D} \leq \frac{L}{D} \leq \frac{\delta}{100 \cdot q_v}$.

Finally, since $D - L - h = L \geq q_{\text{Setup}} + q_v^2/\delta$, we can conclude with the following by Lemma 6.4 and Corollary 6.3.

$$\epsilon(\lambda, D, 0, D - L) \geq \epsilon'(\lambda, D', 0, L) \geq 1 - 6\sqrt{\delta} > \frac{\delta}{100q_v}.$$

7 XFS with grinding

XFS-with-grinding compiles a relativized Σ -protocol into a non-interactive argument by composing a strong PoW, which rules out diagonalization attacks, with a non-amortizing PoW, which supports grinding. Theorem 7.2, instantiated in Corollary 7.3, is the formal version of Theorem 2, and Section 2.2 gives the intuition.

We use PoWs without setup: the prover and the verifier take the difficulty parameter D as input, the puzzle space is $\{0, 1\}^\lambda$, and the puzzle generator samples uniformly from it. The definitions of the PoW properties change accordingly. All tuple encodings below are injective, with separate tags for the two puzzle derivations and the Fiat–Shamir challenge, and the algorithms of PoW_A , the algorithms of PoW_B , and the padder use three further disjoint oracle domains. We omit the tags from the notation. In the challenge query, the tag follows the pad and is part of γ in Definition 3.5.

Let $\Pi = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ be a relativized Σ -protocol as in Definition 3.3, and let PoW_A and PoW_B have difficulties D_A and D_B . The compiled argument is as follows.

Construction 7.1 (XFS-with-grinding). The oracle is $f \in \mathcal{O}(\lambda)$.

- $\mathcal{I}^f(C)$: Output $\psi \leftarrow \mathbf{I}^f(C)$.
- $\mathcal{P}^f((D_A, D_B), C, \mathbb{X}, \mathbb{W})$: Compute $\psi \leftarrow \mathbf{I}^f(C)$ and $(m_1, \text{aux}) \leftarrow \mathbf{P}^f(C, \mathbb{X}, \mathbb{W})$. Then compute

$$\begin{aligned} z_A &\leftarrow f(\psi, \mathbb{X}, m_1), & s_A &\leftarrow \mathbf{P}_A^f(D_A, z_A), \\ z_B &\leftarrow f(\psi, \mathbb{X}, m_1, s_A), & s_B &\leftarrow \mathbf{P}_B^f(D_B, z_B), \\ \tau &\leftarrow \text{Padder}^f(\psi, \mathbb{X}, m_1, s_A, s_B), & \rho &\leftarrow f(\tau, \psi, \mathbb{X}, m_1, s_A, s_B). \end{aligned}$$

Run $m_2 \leftarrow \mathbf{P}^f(\text{aux}, \rho)$ and output $\pi := (m_1, s_A, s_B, m_2)$.

- $\mathcal{V}^f((D_A, D_B), \psi, \mathbb{X}, \pi)$: Parse $\pi = (m_1, s_A, s_B, m_2)$ and recompute z_A, z_B, τ , and ρ as above. Accept if $\mathbf{V}^f(\psi, \mathbb{X}, m_1, \rho, m_2) = 1$, $\mathbf{V}_A^f(D_A, z_A, s_A) = 1$, and $\mathbf{V}_B^f(D_B, z_B, s_B) = 1$.

Theorem 7.2. *Given the following ingredients:*

- $\mathcal{C}$ is a family of circuits, where q_C is a bound on the number of oracle gates.
- Let $\Pi = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ be a relativized SP in the ROM for $\mathcal{C}$, with knowledge soundness error κ as in Definition 3.3, transcript binding error κ_{tb} , index-knowledge error κ_{I} , and indexer query complexity q_{I} .
- Let $\text{PoW}_A = (\mathbf{P}_A, \mathbf{V}_A)$ be a strong PoW in the ROM with difficulty D_A , verifier query complexity q_{V_A} , and strong soundness error ϵ_A .
- Let $\text{PoW}_B = (\mathbf{P}_B, \mathbf{V}_B)$ be a non-amortizing PoW in the ROM with difficulty D_B , verifier query complexity q_{V_B} , and parameter γ_B in Definition 3.12 for every $k \geq 1$ and every expected query bound.
- Let Padder be a prefix-avoiding padder for $\mathbf{V}$ with query complexity q_{pad} and padder error ϵ_{pad} .

Fix $\lambda, q \in \mathbb{N}$. Then Construction 7.1 satisfies adaptive (straightline) knowledge for $\mathcal{C}$ with error κ_{ARG} such that

$$\begin{aligned} \kappa_{\text{ARG}}(\lambda, q) &\leq \left(\frac{q_B}{D_B} + \gamma_B(\lambda, D_B, q_B) \right) \cdot \kappa(\lambda, q_\Sigma) + 2q_{\text{sim}} \cdot \epsilon_A(\lambda, D_A, q_\Sigma, q_A) \\ &\quad + \epsilon_{\text{pad}}(\lambda, q_{\text{sim}} + q_{\text{I}} + q_{\text{pad}} + 1) + q_{\text{sim}} \cdot \kappa_{\text{I}}(\lambda, q_{\text{sim}}) \\ &\quad + q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_\Sigma) + \frac{q_{\text{sim}}^2}{2^\lambda}, \end{aligned}$$

where $q_{\text{sim}} = q + q_I + q_{\text{pad}} + 3$, $q_{\Sigma} = q_{\text{sim}}(2 + q_{\text{pad}} + q_{V_B})$, $q_A = (q_I + q_C)(1 + q_{V_A})$, and $q_B = q_{\text{sim}}(1 + q_{V_B})$. The preprocessing query bound for both strong-soundness reductions is q_{Σ} .

Corollary 7.3 (Instantiation). *In the setting of Theorem 7.2, let PoW_A be the Merkle-tree PoW of Lemma 3.15 and PoW_B the preimage PoW of Lemma 3.16, so $D_A \geq 2\lambda$ and $D_B \leq 2^\lambda$ are powers of two, and let $q_A \leq D_A/4$. Then*

$$\begin{aligned} \kappa_{\text{ARG}}(\lambda, q) \leq & \frac{q_B}{D_B} \cdot \kappa(\lambda, q_{\Sigma}) + \varepsilon_{\text{pad}}(\lambda, q_{\text{sim}} + q_I + q_{\text{pad}} + 1) + q_{\text{sim}} \cdot \kappa_I(\lambda, q_{\text{sim}}) \\ & + q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_{\Sigma}) + \frac{2q_{\text{sim}}(q_{\Sigma} + q_A + 1) + q_{\text{sim}}^2}{2^\lambda}, \end{aligned}$$

with q_{sim} , q_{Σ} , q_A , and q_B as in Theorem 7.2. The last term is negligible when q , q_I , q_{pad} , and D_A are polynomial in λ .

Proof. Since $q_A \leq D_A/4$, Lemma 3.15 gives $\epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) \leq (q_{\Sigma} + q_A + 1)/2^\lambda$. By Lemma 3.16, $\gamma_B = 0$. Substituting both bounds into Theorem 7.2 gives the claim. $\square$

Section 7.1 proves Theorem 7.2 from Lemmas 7.4 to 7.8, which the remaining subsections prove.

7.1 Proof of Theorem 7.2

We use the domain separation specified above. In particular, neither PoW verifier nor any execution of the padder queries a challenge point.

We first complete the transcript of the malicious prover. After it outputs $(C, \mathbb{x}, \pi)$, compute $\psi \leftarrow \mathbf{I}^f(C)$ and the two puzzles, pad, and challenge used in verification, appending the queries and answers to its transcript. This does not change the prover's output and uses at most $q_I + q_{\text{pad}} + 3$ additional queries. In the rest of the proof, $\tilde{\mathcal{P}}$ and $\text{tr}_{\tilde{\mathcal{P}}}$ denote this completed prover and transcript, with query bound q_{sim} . The extractor performs the same completion when given the original transcript.

We define a deterministic recognizer oracle algorithm Rec . By definition, this algorithm makes at most $q_{\text{Rec}} := 1 + q_{\text{pad}} + q_{V_B}$ queries. The recognizer Rec takes a query q as input.

$\text{Rec}^f(q)$:

1. Parse $q = (\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$. If it cannot be parsed this way, return $\perp$.
2. Compute $\tau' \leftarrow \text{Padder}^f(\psi, \mathbb{x}, m_1, s_A, s_B)$ and check if $\tau' = \tau$.
3. Compute $z_B \leftarrow f(\psi, \mathbb{x}, m_1, s_A)$. Check if $V_B^f(D_B, z_B, s_B) = 1$.
4. Output 1 if all checks pass and 0 otherwise.

We define a transcript-augmenting algorithm Aug as follows. The algorithm Aug takes a transcript tr_i as input.

$\text{Aug}^f(\text{tr}_i)$:

1. Initialize an empty transcript tr_f . For each query-answer pair in tr_i , do the following.
 - If the current query is a first occurrence query $q = (\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$, run $b \xleftarrow{\text{tr}_{\text{Rec}}} \text{Rec}^f(q)$ and append tr_{Rec} to tr_f .
 - Append the query-answer pair to tr_f .
2. Output tr_f .

Let $\mathbf{E}$ be the SP extractor for $(\mathbf{I}, \mathbf{P}, \mathbf{V})$. We define the argument extractor $\mathcal{E}$ as follows. The argument extractor $\mathcal{E}$ of the compiled NARK receives as input a circuit C , an instance $\mathbb{x}$, an argument string π , and a query-answer transcript $\text{tr}_{\mathcal{P}}$.

$\mathcal{E}^f(C, \mathbb{x}, \pi, \text{tr}_{\mathcal{P}})$:

1. Parse $\pi = (m_1, s_A, s_B, m_2)$.
2. Complete $\text{tr}_{\mathcal{P}}$ as above and get the augmented transcript $\text{tr} \leftarrow \text{Aug}^f(\text{tr}_{\mathcal{P}})$.
3. Run the SP extractor to compute the witness $\mathbb{w} \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr})$.
4. Output $\mathbb{w}$.

The constructed $\mathcal{E}^f(C, \mathbb{x}, \pi, \text{tr}_{\mathcal{P}})$ makes at most $q_{\mathcal{E}}(\lambda, q) \leq q_I + q_{\text{pad}} + 3 + q_{\text{sim}} q_{\text{Rec}}$ queries.

Let Exp be the following experiment. It runs the knowledge experiment of Definition 3.4 for Construction 7.1 and $\tilde{\mathcal{P}}$, and records the quantities that $\mathcal{V}$ recomputes:

$$\begin{aligned} f &\leftarrow \mathcal{O}(\lambda), & (C, \mathbb{x}, \pi := (m_1, s_A, s_B, m_2)) &\xleftarrow{\text{tr}_{\mathcal{P}}} \tilde{\mathcal{P}}^f, \\ \psi &\leftarrow \mathbf{I}^f(C), & z_A &\leftarrow f(\psi, \mathbb{x}, m_1), \quad z_B \leftarrow f(\psi, \mathbb{x}, m_1, s_A), \\ \tau &\leftarrow \text{Padder}^f(\psi, \mathbb{x}, m_1, s_A, s_B), & \rho &\leftarrow f(\tau, \psi, \mathbb{x}, m_1, s_A, s_B), \\ \text{tr} &\leftarrow \text{Aug}^f(\text{tr}_{\mathcal{P}}), & \mathbb{w} &\leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}). \end{aligned}$$

We call $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$ the *critical query*; its answer is the Fiat–Shamir challenge ρ . We write $\Pr_{\text{Exp}}[\cdot]$ and $\mathbb{E}_{\text{Exp}}[\cdot]$ for probabilities and expectations over Exp . Let $\mathbf{E}_{\text{Fail}}$ be the event that the extractor fails while $\mathcal{V}$ accepts; by the definition of $\mathcal{V}$,

$$\begin{aligned} \mathbf{E}_{\text{Fail}} : \quad & C \in \mathcal{C} \wedge C^f(\mathbb{x}, \mathbb{w}) = 0 \wedge \mathbf{V}^f(\psi, \mathbb{x}, m_1, \rho, m_2) = 1 \\ & \wedge \mathbf{V}_A^f(D_A, z_A, s_A) = 1 \wedge \mathbf{V}_B^f(D_B, z_B, s_B) = 1. \end{aligned}$$

It suffices to bound $\Pr_{\text{Exp}}[\mathbf{E}_{\text{Fail}}]$. Define the following events and notation.

- Define $\mathbf{E}_{\text{PoW}}$ as the event where either $\mathbf{I}^f$ or C^f queries the critical query $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$.
- Define $\mathbf{E}_{\text{Pad}}$ as the event where $\mathbf{V}^f$ or Padder^f queries the critical query $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$. By domain-separation, Rec^f never queries a challenge point: it only runs Padder , derives z_B , and runs $\mathbf{V}_B$.
- For a proof transcript $\text{tr}_{\mathcal{P}}$, define $Q(\text{tr}_{\mathcal{P}})$ to be the list of first-occurrence queries

$$q = (\tau, \psi, \mathbb{x}, m_1, s_A, s_B),$$

for which $\text{Rec}^f(q) = 1$, retaining only the first query for each accepted pair (z_B, s_B) . The list is ordered by first occurrence. Let $N(\text{tr}_{\mathcal{P}}) := |Q(\text{tr}_{\mathcal{P}})|$. Whether a query is added to this list is determined before its answer is given.

- Define $\mathbf{E}_{\text{coll}}$ to be the event that two distinct inputs $(\psi, \mathbb{x}, m_1, s_A)$ deriving z_B receive the same answer, among the inputs queried by $\tilde{\mathcal{P}}$ or derived for its first-occurrence challenge queries.
- Let tr_A be the prefix of $\text{tr}_{\mathcal{P}}$ before the first occurrence of the query $(\psi, \mathbb{x}, m_1)$, and let tr_* be the prefix before the first occurrence of $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$ in $\text{tr}_{\mathcal{P}}$. Let $\text{tr}_{\text{Rec},*}$ be the recognizer transcript for the critical query, and define $\text{tr}_{\text{Aug},*} := \text{Aug}^f(\text{tr}_*) \parallel \text{tr}_{\text{Rec},*}$. Define $\mathbf{E}_{\text{XFS}}$ to be the event that all of the following conditions hold.
 - Both query $(\psi, \mathbb{x}, m_1)$ and query $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$ occur in $\text{tr}_{\mathcal{P}}$, and the former occurs before the latter.
 - $\mathbf{E}_I(\psi, \text{tr}_A) = \mathbf{E}_I(\psi, \text{tr}_*) = C$.
 - $\mathbf{E}(C, \mathbb{x}, m_1, \text{Aug}^f(\text{tr}_A)) = \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_{\text{Aug},*}) = \mathbf{E}(C, \mathbb{x}, m_1, \text{Aug}^f(\text{tr}_{\mathcal{P}}))$

Then, by the law of total probability and union bound, we have

$$\Pr_{\text{Exp}}[\mathbf{E}_{\text{Fail}}] \leq \Pr_{\text{Exp}}[\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_{\text{XFS}} \wedge \bar{\mathbf{E}}_{\text{PoW}} \wedge \bar{\mathbf{E}}_{\text{Pad}} \wedge \bar{\mathbf{E}}_{\text{coll}}]$$

$$\begin{aligned}
& + \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{PoW}}] + \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{Pad}}] \\
& + \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge \bar{E}_{\text{XFS}}] + \Pr_{\text{Exp}} [E_{\text{coll}}].
\end{aligned}$$

The following lemmas, proved in the next subsections, bound these terms.

Lemma 7.4. $\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge \bar{E}_{\text{PoW}} \wedge \bar{E}_{\text{Pad}} \wedge \bar{E}_{\text{coll}}] \leq \mathbb{E}_{\text{Exp}} [N(\text{tr}_{\mathcal{P}})] \cdot \kappa(\lambda, q_{\Sigma}).$

Lemma 7.5. $\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{PoW}}] \leq q_{\text{sim}} \cdot \epsilon_A(\lambda, D_A, q_{\Sigma}, q_A).$

Lemma 7.6. $\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{Pad}}] \leq \varepsilon_{\text{pad}}(\lambda, q_{\text{sim}} + q_{\text{I}} + q_{\text{pad}} + 1).$

Lemma 7.7.

$$\mathbb{E}_{\text{Exp}} [N(\text{tr}_{\mathcal{P}})] \leq \frac{q_B}{D_B} + \gamma_B(\lambda, D_B, q_B), \quad \Pr_{\text{Exp}} [E_{\text{coll}}] \leq \frac{q_{\text{sim}}^2}{2^\lambda},$$

where $q_B = q_{\text{sim}}(1 + q_{V_B})$ and $q_{\text{sim}} = q + q_{\text{I}} + q_{\text{pad}} + 3$. Moreover, on $\bar{E}_{\text{coll}}$, the list $Q(\text{tr}_{\mathcal{P}})$ contains every recognized first-occurrence query.

Lemma 7.8.

$$\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge \bar{E}_{\text{XFS}}] \leq q_{\text{sim}} \cdot \epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) + q_{\text{sim}} \cdot \kappa_{\text{I}}(\lambda, q_{\text{sim}}) + q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_{\Sigma}).$$

Combining their bounds, and substituting the bound on $\mathbb{E}_{\text{Exp}} [N(\text{tr}_{\mathcal{P}})]$ from Lemma 7.7, gives

$$\begin{aligned}
\kappa_{\text{ARG}}(\lambda, q) & \leq \left(\frac{q_B}{D_B} + \gamma_B(\lambda, D_B, q_B) \right) \cdot \kappa(\lambda, q_{\Sigma}) + 2q_{\text{sim}} \cdot \epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) \\
& + \varepsilon_{\text{pad}}(\lambda, q_{\text{sim}} + q_{\text{I}} + q_{\text{pad}} + 1) + q_{\text{sim}} \cdot \kappa_{\text{I}}(\lambda, q_{\text{sim}}) \\
& + q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_{\Sigma}) + \frac{q_{\text{sim}}^2}{2^\lambda}.
\end{aligned}$$

7.2 Proof of Lemma 7.4

Split the malicious prover's execution at the i -th query in $Q(\text{tr}_{\mathcal{P}})$. The two phases together make at most $q_{\text{sim}}(1 + q_{\text{Rec}})$ queries. For a fixed index $i \in [q_{\text{sim}}]$, the algorithm $\tilde{\mathcal{P}}_{0 \rightarrow i}^f$ runs the $\tilde{\mathcal{P}}$ up to the i -th query in $Q(\text{tr}_{\mathcal{P}})$, and $\tilde{\mathcal{P}}_{i \rightarrow q_{\text{sim}}}$ continues the execution from that point to the end.

• $\tilde{\mathcal{P}}_{0 \rightarrow i}^f$:

1. Initialize a counter $\text{ctr} = 0$ and an empty set S .
2. Simulate $\tilde{\mathcal{P}}^f$, storing all of its oracle query-response pairs in $\text{tr}_{0 \rightarrow i}$. Whenever it makes a first-occurrence query $q = (\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$, check if $\text{Rec}^f(q) = 1$ before answering the query. If so, obtain z_B from this recognizer execution. If $(z_B, s_B) \notin S$, add this pair to S and increment ctr by 1.
3. If $\tilde{\mathcal{P}}$ halts while $\text{ctr} < i$, let $\hat{q} = \perp$. Otherwise, once $\text{ctr} = i$, halt before answering the query. Store this i -th query $(\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$ as $\hat{q}$.
4. Output the i -th query $\hat{q}$, the pure transcript prefix of $\tilde{\mathcal{P}}^f$ stored in $\text{tr}_{0 \rightarrow i}$, and its state $\text{aux}_{\tilde{\mathcal{P}}}$.

• $\tilde{\mathcal{P}}_{i \rightarrow q_{\text{sim}}}^f(\text{aux}_{\tilde{\mathcal{P}}}, \rho)$:

1. Simulate $(C, \mathbb{X}, (m_1, s_A, s_B, m_2)) \xleftarrow{\text{tr}_{i \rightarrow q_{\text{sim}}}} \tilde{\mathcal{P}}^f$ using $\text{aux}_{\tilde{\mathcal{P}}}$ to continue the execution from the i -th query in $Q(\text{tr}_{\mathcal{P}})$ given oracle response ρ .
2. Output $(C, \mathbb{X}, (m_1, s_A, s_B, m_2), \text{tr}_{i \rightarrow q_{\text{sim}}})$.

We use the algorithms above to construct a malicious SP prover as follows. By definition, the query bound of this SP prover is $q_\Sigma := q_{\text{sim}}(1 + q_{\text{Rec}})$.

- First SP prover message: $\tilde{\mathbf{P}}^f$
 1. Run $(\hat{q}, \text{tr}_{0 \rightarrow i}, \text{aux}_{\tilde{\mathcal{P}}}) \xleftarrow{\text{tr}} \tilde{\mathcal{P}}_{0 \rightarrow i}^f$.
 2. Parse $\hat{q} = (\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$. If it cannot be parsed this way, output $\perp$ in every component.
 3. Compute $C \leftarrow \mathbf{E}_I(\psi, \text{tr}_{0 \rightarrow i})$.
 4. Output $(C, \mathbb{x}, m_1, \text{aux} = (i, \hat{q}, \text{aux}_{\tilde{\mathcal{P}}}))$.
- Second SP prover message: $\mathbf{P}^f(\text{aux} = (i, \hat{q}, \text{aux}_{\tilde{\mathcal{P}}}), \rho)$
 1. If $\text{aux} = \perp$, output $\perp$. Otherwise, set $\mu = \{(\hat{q}, \rho)\}$.
 2. Run $(C', \mathbb{x}', (m'_1, s'_A, s'_B, m_2), \text{tr}_{i \rightarrow q_{\text{sim}}}) \leftarrow \tilde{\mathcal{P}}_{i \rightarrow q_{\text{sim}}}^{f[\mu]}(\text{aux}_{\tilde{\mathcal{P}}}, \rho)$. Answer every query to $\hat{q}$ using the stored value ρ , and forward all other queries to f .
 3. Output m_2 .

Define $\mathbf{E}_{\text{reach}, i}$ to be the event that $\hat{q} \neq \perp$. This event is determined entirely by the pre-challenge view view_1 , and

$$\Pr[\mathbf{E}_{\text{reach}, i}] = \Pr_{\text{Exp}}[N(\text{tr}_{\mathcal{P}}) \geq i].$$

We average the conditional knowledge bound over the pre-challenge views for which $\mathbf{E}_{\text{reach}, i}$ occurs. Conditional knowledge soundness of Π gives

$$\kappa(\lambda, q_\Sigma) \cdot \Pr[\mathbf{E}_{\text{reach}, i}] \geq \Pr \left[\begin{array}{l} \mathbf{E}_{\text{reach}, i} \\ \wedge C \in \mathcal{C} \\ \wedge C^f(\mathbb{x}, \mathbb{w}) = 0 \\ \wedge \mathbf{V}^f(\psi, \mathbb{x}, m_1, \rho, m_2) = 1 \end{array} \middle| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, m_1, \text{aux}) \xleftarrow{\text{tr}} \tilde{\mathbf{P}}^f \\ \text{view}_1 \leftarrow (C, \mathbb{x}, m_1, \text{aux}, \text{tr}) \\ \rho \leftarrow \{0, 1\}^\lambda \\ m_2 \leftarrow \tilde{\mathbf{P}}^f(\text{aux}, \rho) \\ \mathbb{w} \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}) \\ \psi \leftarrow \mathbf{I}^f(C) \end{array} \right].$$

Note that by the end of the first phase, $\tilde{\mathcal{P}}$ has never received $f(\hat{q})$. The recognizer does not query $\hat{q}$ because of domain-separation. Thus, $\hat{q}$ is fresh in the complete first-phase transcript tr , and its oracle response is uniform even after fixing view_1 . The continuation answers this query locally with ρ , without querying the real oracle at $\hat{q}$.

Define $\mathbf{E}_{\text{crit}, i}$ to be the event that the critical query of the final proof has rank i in $Q(\text{tr}_{\mathcal{P}})$. On $\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_{\text{XFS}} \wedge \bar{\mathbf{E}}_{\text{coll}}$, the recognizer accepts the critical query, so it occurs in $Q(\text{tr}_{\mathcal{P}})$ by Lemma 7.7. If $\mathbf{E}_{\text{crit}, i}$ occurs, the constructed SP prover embeds the challenge ρ at exactly the right query point in the execution of $\tilde{\mathcal{P}}$. Moreover, the fields parsed from $\hat{q}$ are consistent with the output fields of $\tilde{\mathcal{P}}_{i \rightarrow q_{\text{sim}}}^{f[\mu]}$.

On $\bar{\mathbf{E}}_{\text{PoW}} \wedge \bar{\mathbf{E}}_{\text{pad}}$, none of the SP indexer, the extracted circuit, or the SP verifier queries the critical query. The PoW verifiers and the padder do not query the critical query because of domain-separation. Therefore, replacing $f(\hat{q})$ by the independent uniform value ρ gives the same distribution of the simulated execution. On these events, the indexer, circuit, and verifier have the same outputs under f and $f[\mu]$.

On $\mathbf{E}_{\text{XFS}} \wedge \mathbf{E}_{\text{crit}, i}$, the circuit extracted from $\text{tr}_{0 \rightarrow i}$ is the same as the eventual circuit output by $\tilde{\mathcal{P}}$. Moreover, on $\mathbf{E}_{\text{crit}, i}$, the first-phase transcript is

$$\text{tr} = \text{Aug}^f(\text{tr}_*) \parallel \text{tr}_{\text{Rec}, *} = \text{tr}_{\text{Aug}, *}.$$

This transcript includes the recognizer queries for $\hat{q}$ and excludes its answer. By the definition of $\mathbf{E}_{\text{XFS}}$, we have

$$\mathbf{E}(C, \mathbb{x}, m_1, \text{tr}) = \mathbf{E}(C, \mathbb{x}, m_1, \text{Aug}^f(\text{tr}_{\mathcal{P}})).$$

Restricting the SP failure event to these executions gives

$$\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge \bar{E}_{\text{PoW}} \wedge \bar{E}_{\text{Pad}} \wedge \bar{E}_{\text{coll}} \wedge E_{\text{crit},i}] \leq \kappa(\lambda, \mathbf{q}_\Sigma) \cdot \Pr_{\text{Exp}} [N(\text{tr}_{\mathcal{P}}) \geq i].$$

Finally, summing over all possible ranks of the critical query, we conclude that

$$\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge \bar{E}_{\text{PoW}} \wedge \bar{E}_{\text{Pad}} \wedge \bar{E}_{\text{coll}}] \leq \kappa(\lambda, \mathbf{q}_\Sigma) \sum_{i=1}^{\mathbf{q}_{\text{sim}}} \Pr_{\text{Exp}} [N(\text{tr}_{\mathcal{P}}) \geq i] = \mathbb{E}_{\text{Exp}} [N(\text{tr}_{\mathcal{P}})] \cdot \kappa(\lambda, \mathbf{q}_\Sigma).$$

7.3 Proof of Lemma 7.5

Split the malicious prover's execution at its j -th query. The two phases together make at most $\mathbf{q}_{\text{sim}}(1 + \mathbf{q}_{\text{Rec}})$ queries. For an index $j \in [\mathbf{q}_{\text{sim}}]$, the algorithm $\tilde{\mathcal{P}}_{0 \rightarrow j}^f$ runs the $\tilde{\mathcal{P}}$ up to the j -th query, and $\tilde{\mathcal{P}}_{j \rightarrow \mathbf{q}_{\text{sim}}}$ continues the execution from that point to the end.

- $\tilde{\mathcal{P}}_{0 \rightarrow j}^f$:
 1. Simulate $\tilde{\mathcal{P}}^f$ up to the j -th query. If $\tilde{\mathcal{P}}$ finishes running before then, output $\perp$. Whenever it makes a first-occurrence query $q = (\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$, run $\text{Rec}^f(q)$ before answering the query.
 2. Stop before answering the j -th query, after running its recognizer if it has the above form and is a first occurrence. Output the j -th query $\hat{q}$, the pure transcript prefix $\text{tr}_{0 \rightarrow j}$ and its state $\text{aux}_{\tilde{\mathcal{P}}}$.
- $\tilde{\mathcal{P}}_{j \rightarrow \mathbf{q}_{\text{sim}}}^f(\text{aux}_{\tilde{\mathcal{P}}}, u)$:
 1. Simulate $(C, \mathbb{X}, (m_1, s_A, s_B, m_2)) \leftarrow \tilde{\mathcal{P}}^f$ using $\text{aux}_{\tilde{\mathcal{P}}}$ to continue the execution from the j -th query given oracle response u . Whenever it makes a subsequent first-occurrence query $q = (\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$, run $\text{Rec}^f(q)$ before answering the query.
 2. Output $(C, \mathbb{X}, (m_1, s_A, s_B, m_2))$.

The first-phase transcript at a puzzle query is $\text{Aug}^f(\text{tr}_{0 \rightarrow j})$. At a first-occurrence critical query it additionally contains the transcript of the recognizer on the selected query, as in the definition of E_{XFS} .

We construct adversary $A = (A_1, A_2)$ against the strong soundness of PoW_A .

$A_1^f(D_A)$:

1. Sample $j \leftarrow [\mathbf{q}_{\text{sim}}]$.
2. Run $((\psi, \mathbb{X}, m_1), \text{tr}_{0 \rightarrow j}, \text{aux}_{\tilde{\mathcal{P}}}) \xleftarrow{\text{tr}} \tilde{\mathcal{P}}_{0 \rightarrow j}^f$. Abort if the j -th query cannot be parsed in this way or has occurred before.
3. Compute $C \leftarrow \mathbf{E}_I(\psi, \text{tr}_{0 \rightarrow j})$. Abort if $C \notin \mathcal{C}$.
4. Compute $\mathbb{W} \leftarrow \mathbf{E}(C, \mathbb{X}, m_1, \text{tr})$.
5. Output $\text{aux} = (C, \psi, \mathbb{X}, m_1, \mathbb{W})$.

Given a randomly sampled puzzle $z_A \leftarrow \{0, 1\}^\lambda$, the online adversary A_2 solves the puzzle as follows.

$A_2^f(\text{aux} = (C, \psi, \mathbb{X}, m_1, \mathbb{W}), z_A)$:

1. Define $\mu := \{((\psi, \mathbb{X}, m_1), z_A)\}$.
2. Run $\mathbf{I}^{f[\mu]}(C)$ and $C^{f[\mu]}(\mathbb{X}, \mathbb{W})$, recording their query-answer transcripts tr_I and tr_C .
3. For each query q in $\text{tr}_I \cup \text{tr}_C$ do the following:
 - (a) Parse $q = (\tau, \psi', \mathbb{X}', m'_1, s_A, s_B)$. If it cannot be parsed this way, move on to the next query.
 - (b) If $V_A^f(D_A, z_A, s_A) = 1$, output s_A and halt.

4. Output $\perp$.

The preprocessing stage makes at most $q_{\text{sim}}(1 + q_{\text{Rec}}) = q_{\Sigma}$ queries; both extractors use only their input transcripts. By construction, A_2 makes at most $(q_I + q_C)(1 + q_{V_A}) = q_A$ queries. Strong soundness of PoW_A gives the following bound. Whenever we write out the steps of A_1 below, we retain its parsing and first-occurrence checks.

$$\begin{aligned} \epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) &\geq \Pr \left[V_A^f(D_A, z_A, s_A) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{aux} \leftarrow A_1^f(D_A) \\ z_A \leftarrow \{0, 1\}^\lambda \\ s_A \leftarrow A_2^f(\text{aux}, z_A) \end{array} \right] \\ &\geq \Pr \left[\begin{array}{l} C \in \mathcal{C} \\ \wedge \exists (\tau, \psi', \mathbb{X}', m'_1, s_A, s_B) \in \text{tr}_I \cup \text{tr}_C \\ \text{s.t. } V_A^f(D_A, z_A, s_A) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ j \leftarrow [q_{\text{sim}}] \\ ((\psi, \mathbb{X}, m_1), \text{tr}_{0 \rightarrow j}, \text{aux}_{\tilde{\mathcal{P}}}) \xleftarrow{\text{tr}} \tilde{\mathcal{P}}_{0 \rightarrow j}^f \\ z_A \leftarrow \{0, 1\}^\lambda \\ C \leftarrow \mathbf{E}_I(\psi, \text{tr}_{0 \rightarrow j}) \\ \mathbb{W} \leftarrow \mathbf{E}(C, \mathbb{X}, m_1, \text{tr}) \\ \mu = \{((\psi, \mathbb{X}, m_1), z_A)\} \\ y \xleftarrow{\text{tr}_I} \mathbf{I}^{f[\mu]}(C) \\ b \xleftarrow{\text{tr}_C} C^{f[\mu]}(\mathbb{X}, \mathbb{W}) \end{array} \right]. \end{aligned}$$

Since the selected query is a first occurrence, $\tilde{\mathcal{P}}_{0 \rightarrow j}$ has not received the oracle response of this query by the end of its execution. By domain separation, neither the recognizer queries nor V_A query this point. Therefore, we can replace the uniformly random z_A with the actual oracle response, without changing the distribution. This gives

$$\epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) \geq \Pr \left[\begin{array}{l} C \in \mathcal{C} \\ \wedge \exists (\tau, \psi', \mathbb{X}', m'_1, s_A, s_B) \in \text{tr}_I \cup \text{tr}_C \\ \text{s.t. } V_A^f(D_A, z_A, s_A) = 1 \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ j \leftarrow [q_{\text{sim}}] \\ ((\psi, \mathbb{X}, m_1), \text{tr}_{0 \rightarrow j}, \text{aux}_{\tilde{\mathcal{P}}}) \xleftarrow{\text{tr}} \tilde{\mathcal{P}}_{0 \rightarrow j}^f \\ z_A \leftarrow f(\psi, \mathbb{X}, m_1) \\ C \leftarrow \mathbf{E}_I(\psi, \text{tr}_{0 \rightarrow j}) \\ \mathbb{W} \leftarrow \mathbf{E}(C, \mathbb{X}, m_1, \text{tr}) \\ y \xleftarrow{\text{tr}_I} \mathbf{I}^f(C) \\ b \xleftarrow{\text{tr}_C} C^f(\mathbb{X}, \mathbb{W}) \end{array} \right].$$

On E_{XFS} , $\tilde{\mathcal{P}}$ makes the query $(\psi, \mathbb{X}, m_1)$. Let j_A be the index of its first occurrence in the completed transcript. If the randomly selected j is the correct index, then the constructed adversary embeds the puzzle at exactly the right query point in the execution of $\tilde{\mathcal{P}}$.

On E_{XFS} , the circuit extracted from the pure prefix tr_A is the same as the circuit output by $\tilde{\mathcal{P}}$. The witness extracted from $\text{Aug}^f(\text{tr}_A)$ is the same as the witness extracted from $\text{Aug}^f(\text{tr}_{\mathcal{P}})$. On E_{PoW} , the valid critical query appears in $\text{tr}_I \cup \text{tr}_C$. The online adversary performs only the indexer and circuit evaluations and the candidate verifications. Therefore, on $E_{\text{XFS}} \wedge E_{\text{PoW}}$ and $j = j_A$, the online adversary finds the critical query in $\text{tr}_I \cup \text{tr}_C$ and outputs an accepted solution. Since j is uniform and independent of Exp , the event $j = j_A$ has probability $1/q_{\text{sim}}$ conditioned on every outcome of Exp , and hence

$$\epsilon_A(\lambda, D_A, q_{\Sigma}, q_A) \geq \frac{1}{q_{\text{sim}}} \cdot \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{PoW}}].$$

7.4 Proof of Lemma 7.6

Consider the following adversary and machine for the prefix-avoidance experiment.

- A^f :
 1. Run $(C, \mathbb{X}, (m_1, s_A, s_B, m_2)) \xleftarrow{\text{tr}_P} \tilde{\mathcal{P}}^f$.
 2. Compute $\psi \leftarrow \mathbf{I}^f(C)$.
 3. Compute $\tau \leftarrow \text{Padder}^f(\psi, \mathbb{X}, m_1, s_A, s_B)$.
 4. Compute $\rho \leftarrow f(\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$.
 5. Output $\gamma = (\psi, \mathbb{X}, m_1, s_A, s_B)$ and $m = (\psi, \mathbb{X}, m_1, \rho, m_2)$.
- $M^f(m = (\psi, \mathbb{X}, m_1, \rho, m_2))$:
 1. Run $\mathbf{V}^f(\psi, \mathbb{X}, m_1, \rho, m_2)$.

By domain separation, no execution of Padder or Rec queries a critical query. The adversary A and the machine M together perform the steps of Exp that precede the verifier checks, with tr the transcript of Padder and tr_M that of $\mathbf{V}$, so the event $(\tau, \gamma) \in \text{tr} \cup \text{tr}_M$ is E_{Pad} . By construction, A makes at most $q_{\text{sim}} + q_{\text{I}} + q_{\text{pad}} + 1$ queries. Prefix avoidance gives

$$\begin{aligned}
 \varepsilon_{\text{pad}}(\lambda, q_{\text{sim}} + q_{\text{I}} + q_{\text{pad}} + 1) &\geq \Pr \left[(\tau, \gamma) \in \text{tr} \cup \text{tr}_M \left| \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ (\gamma, m) \leftarrow A^f \\ \tau \xleftarrow{\text{tr}} \text{Padder}^f(\gamma) \\ y \xleftarrow{\text{tr}_M} M^f(m) \end{array} \right. \right] \\
 &= \Pr_{\text{Exp}} [E_{\text{Pad}}] \\
 &\geq \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_{\text{XFS}} \wedge E_{\text{Pad}}].
 \end{aligned}$$

7.5 Proof of Lemma 7.7

We construct an adversary A_B for the non-amortization of PoW_B as follows.

$A_B^f(D_B)$:

- Initialize empty set S .
- Simulate the completed $\tilde{\mathcal{P}}^f$; simulate all non-B oracle domains locally and forward only B-domain queries to its real oracle. Whenever it issues a first occurrence query

$$q = (\tau, \psi, \mathbb{X}, m_1, s_A, s_B),$$

do the following:

- * Compute $z_B \leftarrow f(\psi, \mathbb{X}, m_1, s_A)$ using the local simulation.
- * If $(z_B, s_B) \in S$, continue to the next query-answer pair. Otherwise, check whether

$$V_B^f(D_B, z_B, s_B) = 1.$$

If so, add (z_B, s_B) to S .

- Output S .

By domain-separation, the simulation has the same distribution as the completed execution of $\tilde{\mathcal{P}}^f$. By construction, A_B outputs at most q_{sim} pairs and verifies each of them. It forwards at most q_{sim} B-domain queries and runs V_B at most q_{sim} times. Therefore, it makes at most $q_B = q_{\text{sim}}(1 + q_{V_B})$ queries, including all verification queries. By the setup-free form of Definition 3.12, applied with $k = q_{\text{sim}}$ and query bound q_B , we have

$$\mathbb{E}_{\text{Exp}} [N(\text{tr}_{\mathcal{P}})] \leq \mathbb{E} [|S|] \leq \frac{q_B}{D_B} + \gamma_B(\lambda, D_B, q_B).$$

Every query in $Q(\text{tr}_{\mathcal{P}})$ contains an accepted puzzle–solution pair, and these pairs are distinct by the definition of $Q(\text{tr}_{\mathcal{P}})$. The collector stores all such pairs. Thus, $N(\text{tr}_{\mathcal{P}}) \leq |S|$, which proves the first inequality.

In the simulation above, there are at most q_{sim} inputs $(\psi, \mathbb{x}, m_1, s_A)$ to the oracle deriving z_B : each is either queried directly by $\tilde{\mathcal{P}}$, or evaluated by the collector for a first occurrence challenge query. These two query shapes are disjoint, and each challenge query causes at most one additional evaluation. Therefore, by the birthday bound,

$$\Pr_{\text{Exp}} [E_{\text{coll}}] \leq \binom{q_{\text{sim}}}{2} 2^{-\lambda} \leq \frac{q_{\text{sim}}^2}{2^\lambda}.$$

Suppose two recognized queries contain the same accepted pair (z_B, s_B) . On $\bar{E}_{\text{coll}}$, their inputs $(\psi, \mathbb{x}, m_1, s_A)$ to the oracle deriving z_B must coincide. Their second solutions also coincide by assumption. The deterministic paddler therefore gives the same pad, so the two queries coincide. Thus, on $\bar{E}_{\text{coll}}$, $Q(\text{tr}_{\mathcal{P}})$ contains every distinct recognized query.

7.6 Proof of Lemma 7.8

Define the following events.

- E_1 : the first occurrence of $(\psi, \mathbb{x}, m_1)$ comes after the first occurrence of $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$ in $\text{tr}_{\mathcal{P}}$.
- E_2 : $\mathbf{E}_I(\psi, \text{tr}_A) \neq C$ or $\mathbf{E}_I(\psi, \text{tr}_*) \neq C$.
- E_3 : at least one of the two prefix witnesses in the definition of E_{XFS} differs from $\mathbf{E}(C, \mathbb{x}, m_1, \text{Aug}^f(\text{tr}_{\mathcal{P}}))$.

Both queries occur in $\text{tr}_{\mathcal{P}}$ by the transcript completion for $\tilde{\mathcal{P}}$. By definition, $\bar{E}_{\text{XFS}}$ is the event where at least one of the above occurs. The union bound gives

$$\Pr_{\text{Exp}} [E_{\text{Fail}} \wedge \bar{E}_{\text{XFS}}] \leq \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_1] + \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_2] + \Pr_{\text{Exp}} [E_{\text{Fail}} \wedge E_3 \wedge \bar{E}_2].$$

We bound each of these three terms below.

Split the malicious prover's execution at its j -th query, without augmenting its transcript. For an index $j \in [q_{\text{sim}}]$, the algorithm $\hat{\mathcal{P}}_{0 \rightarrow j}^f$ runs $\tilde{\mathcal{P}}$ up to the j -th query, and $\hat{\mathcal{P}}_{j \rightarrow q_{\text{sim}}}$ continues the execution from that point to the end.

- $\hat{\mathcal{P}}_{0 \rightarrow j}^f$:
 1. Simulate $\tilde{\mathcal{P}}^f$ up to, but not including, its j -th query. If $\tilde{\mathcal{P}}$ has already finished running, output $\perp$.
 2. Output the j -th query $\hat{q}$ and its state $\text{aux}_{\hat{\mathcal{P}}}$.
- $\hat{\mathcal{P}}_{j \rightarrow q_{\text{sim}}}^f(\text{aux}_{\hat{\mathcal{P}}}, u)$:
 1. Simulate $(C, \mathbb{x}, (m_1, s_A, s_B, m_2)) \leftarrow \tilde{\mathcal{P}}^f$ using $\text{aux}_{\hat{\mathcal{P}}}$ to continue the execution from the j -th query given oracle response u .

2. Output $(C, \mathbb{X}, (m_1, s_A, s_B, m_2))$.

Bounding $\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_1]$. We construct the adversary $A = (A_1, A_2)$ against the strong soundness of PoW_A . The preprocessing adversary A_1 computes the auxiliary information as follows.

$A_1^f(D_A)$:

1. Sample $j \leftarrow [q_{\text{sim}}]$.
2. Simulate $(\hat{q}, \text{aux}_{\hat{p}}) \xleftarrow{\text{tr}} \hat{\mathcal{P}}_{0 \rightarrow j}^f$. Parse $\hat{q} = (\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$. If $\hat{q}$ cannot be parsed this way or $(\psi, \mathbb{X}, m_1)$ has already been queried, abort.
3. Output $\text{aux} \leftarrow s_A$.

Given a randomly sampled puzzle $z_A \leftarrow \{0, 1\}^\lambda$, the online adversary A_2 simply outputs the solution stored in the auxiliary input.

$A_2^f(\text{aux}, z_A)$:

1. Parse $\text{aux} = s_A$. Abort if it cannot be parsed this way.
2. Output s_A .

The preprocessing stage makes at most $q_{\text{sim}} \leq q_\Sigma$ queries. By construction, A_2 makes 0 online queries. Strong soundness of PoW_A gives the following bound.

$$\epsilon_A(\lambda, D_A, q_\Sigma, q_A) \geq \Pr \left[V_A^f(D_A, z_A, s_A) = 1 \mid \begin{array}{l} f \leftarrow \mathcal{O}(\lambda) \\ \text{aux} \leftarrow A_1^f(D_A) \\ z_A \leftarrow \{0, 1\}^\lambda \\ s_A \leftarrow A_2^f(\text{aux}, z_A) \end{array} \right].$$

There is $1/q_{\text{sim}}$ probability that $\hat{q}$ is the first occurrence of the final critical query. On $\mathbf{E}_1$, $\tilde{\mathcal{P}}$ has never obtained the puzzle z_A before fixing its solution s_A . The query deriving this puzzle is fresh at this point, and V_A does not query it by domain separation. Therefore, we can replace the randomly sampled z_A with the actual oracle response, without changing the distribution. On $\mathbf{E}_{\text{Fail}}$, the selected solution is accepted. It follows that

$$\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_1] \leq q_{\text{sim}} \cdot \epsilon_A(\lambda, D_A, q_\Sigma, q_A).$$

Bounding $\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_2]$. We construct an adversary against the indexer-knowledge of the Sigma-protocol as follows.

• A^f :

1. Randomly select $j \leftarrow [q_{\text{sim}}]$.
2. Run $(\hat{q}, \text{aux}_{\hat{p}}) \leftarrow \hat{\mathcal{P}}_{0 \rightarrow j}^f$.
3. If $\hat{q}$ has the form $(\psi, \mathbb{X}, m_1)$ or $(\tau, \psi, \mathbb{X}, m_1, s_A, s_B)$, output ψ and $\text{aux} = (j, \hat{q}, \text{aux}_{\hat{p}})$. Otherwise, abort.

• $A^f(\text{aux} = (j, \hat{q}, \text{aux}_{\hat{p}}))$:

1. Compute $u \leftarrow f(\hat{q})$.
2. Run $(C, \mathbb{X}, (m_1, s_A, s_B, m_2)) \leftarrow \hat{\mathcal{P}}_{j \rightarrow q_{\text{sim}}}^f(\text{aux}_{\hat{p}}, u)$.
3. Output C .

By construction, $\mathbf{A}$ makes at most q_{sim} queries. Indexer knowledge soundness bounds the probability that $\mathbf{A}$ outputs a circuit different from the one extracted from ψ :

$$\kappa_{\mathbf{I}}(\lambda, q_{\text{sim}}) \geq \Pr \left[\begin{array}{c} \mathbf{E}_{\mathbf{I}}(\psi, \text{tr}) \neq C \\ \wedge \mathbf{I}^f(C) = \psi \end{array} \middle| \begin{array}{c} f \leftarrow \mathcal{O}(\lambda) \\ (\psi, \text{aux}) \xleftarrow{\text{tr}} \mathbf{A}^f \\ C \leftarrow \mathbf{A}^f(\text{aux}) \end{array} \right].$$

On $\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_2$, at least one of the first puzzle query and the first critical query has a prefix from which the index extractor outputs a different circuit. If j selects either such query, the constructed adversary succeeds. The probability of selecting such a query is at least $1/q_{\text{sim}}$. Therefore,

$$\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_2] \leq q_{\text{sim}} \cdot \kappa_{\mathbf{I}}(\lambda, q_{\text{sim}}).$$

Bounding $\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_3 \wedge \bar{\mathbf{E}}_2]$. For the transcript-binding experiment, $\tilde{\mathbf{P}}$ runs $\bar{\mathcal{P}}_{0 \rightarrow j}$ up to a randomly chosen query, augmenting the transcript as it goes. The second adversary continues the execution and its transcript augmentation to the end.

- $\tilde{\mathbf{P}}^f$:
 1. Sample $j \leftarrow [q_{\text{sim}}]$ at random.
 2. Simulate $(\hat{q}, \text{tr}_{0 \rightarrow j}, \text{aux}_{\tilde{\mathcal{P}}}) \leftarrow \bar{\mathcal{P}}_{0 \rightarrow j}^f$.
 3. If $\hat{q}$ has the form $(\psi, \mathbb{x}, m_1)$ or $(\tau, \psi, \mathbb{x}, m_1, s_A, s_B)$, run $C \leftarrow \mathbf{E}_{\mathbf{I}}(\psi, \text{tr}_{0 \rightarrow j})$. Otherwise, abort.
 4. Output $(C, \mathbb{x}, m_1, \text{aux} = (j, \hat{q}, \text{aux}_{\tilde{\mathcal{P}}}))$.
- $\mathbf{A}^f(\text{aux} = (j, \hat{q}, \text{aux}_{\tilde{\mathcal{P}}}))$:
 1. Compute $u \leftarrow f(\hat{q})$.
 2. Run $\bar{\mathcal{P}}_{j \rightarrow q_{\text{sim}}}^f(\text{aux}_{\tilde{\mathcal{P}}}, u)$.

By construction, $\tilde{\mathbf{P}}$ and $\mathbf{A}$ each make at most $q_{\Sigma} = q_{\text{sim}}(1 + q_{\text{Rec}})$ queries. Transcript binding of Π gives

$$\kappa_{\text{tb}}(\lambda, q_{\Sigma}) \geq \Pr \left[\begin{array}{c} C \in \mathcal{C} \\ \wedge \mathbb{W} \neq \mathbb{W}' \end{array} \middle| \begin{array}{c} f \leftarrow \mathcal{O}(\lambda) \\ (C, \mathbb{x}, m_1, \text{aux}) \xleftarrow{\text{tr}_1} \tilde{\mathbf{P}}^f \\ \perp \xleftarrow{\text{tr}_2} \mathbf{A}^f(\text{aux}) \\ \mathbb{W} \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_1) \\ \mathbb{W}' \leftarrow \mathbf{E}(C, \mathbb{x}, m_1, \text{tr}_1 \parallel \text{tr}_2) \end{array} \right].$$

On $\bar{\mathbf{E}}_2$, selecting either of the two relevant queries makes the circuit output by the first phase equal to the final circuit C . The first-phase transcript is exactly the corresponding augmented prefix in the definition of $\mathbf{E}_{\text{XFS}}$, including the selected recognizer transcript at the critical query. The complete transcript is $\text{Aug}^f(\text{tr}_{\mathcal{P}})$.

On $\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_3 \wedge \bar{\mathbf{E}}_2$, at least one of these two prefixes gives a different extracted witness from the complete transcript. The probability of selecting such a query is at least $1/q_{\text{sim}}$. It follows that

$$\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \mathbf{E}_3 \wedge \bar{\mathbf{E}}_2] \leq q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_{\Sigma}).$$

Combining the three bounds, we conclude that

$$\Pr_{\text{Exp}} [\mathbf{E}_{\text{Fail}} \wedge \bar{\mathbf{E}}_{\text{XFS}}] \leq q_{\text{sim}} \cdot \epsilon_{\mathbf{A}}(\lambda, D_{\mathbf{A}}, q_{\Sigma}, q_{\mathbf{A}}) + q_{\text{sim}} \cdot \kappa_{\mathbf{I}}(\lambda, q_{\text{sim}}) + q_{\text{sim}} \cdot \kappa_{\text{tb}}(\lambda, q_{\Sigma}).$$

Acknowledgments

Ziyi Guan thanks Alessandro Chiesa, Artur Riazanov, Eylon Yogev, and Weiqiang Yuan for helpful discussions related to this work.

AI disclosure

This paper has two main results: the impossibility of a strong and non-amortizing PoW (Theorem 1; formally Theorem 6.1) and the XFS-with-grinding transformation (Theorem 2; formally Theorem 7.2). We used large language models (LLMs) as follows. For the impossibility result, the initial ideas are the authors'. While turning the intuition into a proof, we used an LLM in iterative sessions to clean up the ideas and simplify the argument. The LLM drafted a proof sketch, and the authors wrote the proof. For the XFS-with-grinding result, the ideas and the proofs are the authors' own. Throughout the paper, we used an LLM to polish the text. The authors have verified all technical content and take full responsibility for it.

References

- [AAKTC26] Hamza Abusalah, Karen Azari, Chethan Kamath, Erkan Tairi, and Maximilian von Consbruch. “Impossibility of VDFs in the ROM: The Complete Picture”. In: *EUROCRYPT 2026, Part V*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16545. LNCS. Springer, Cham, May 2026, pp. 3–32. DOI: 10.1007/978-3-032-25330-9_1.
- [ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. “STIR: Reed-Solomon Proximity Testing with Fewer Queries”. In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 380–413. DOI: 10.1007/978-3-031-68403-6_12.
- [ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. “WHIR: Reed-Solomon Proximity Testing with Super-Fast Verification”. In: *EUROCRYPT 2025, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 214–243. DOI: 10.1007/978-3-031-91134-7_8.
- [AY25] Gal Arnon and Eylon Yogev. “Towards a White-Box Secure Fiat-Shamir Transformation”. In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 27–56. DOI: 10.1007/978-3-032-01887-8_2.
- [BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. “Verifiable Delay Functions”. In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.
- [BBHMR19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. “On the (In)security of Kilian-Based SNARGs”. In: *TCC 2019, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. LNCS. Springer, Cham, Dec. 2019, pp. 522–551. DOI: 10.1007/978-3-030-36033-7_20.
- [BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. “Fast Reed-Solomon Interactive Oracle Proofs of Proximity”. In: *ICALP 2018*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl, July 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14.
- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. “Recursive composition and bootstrapping for SNARKS and proof-carrying data”. In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 111–120. DOI: 10.1145/2488608.2488623.

- [BCG25] Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. “Relativized Succinct Arguments in the ROM do not Exist”. In: *TCC 2025, Part I*. Ed. by Benny Applebaum and Huijia (Rachel) Lin. Vol. 16268. LNCS. Springer, Cham, Dec. 2025, pp. 400–416. DOI: 10.1007/978-3-032-12287-2_14.
- [BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. “Scalable Zero Knowledge via Cycles of Elliptic Curves”. In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Berlin, Heidelberg, Aug. 2014, pp. 276–294. DOI: 10.1007/978-3-662-44381-1_16.
- [BFRW25] Benedikt Bünz, Giacomo Fenzi, Ron D. Rothblum, and William Wang. *TensorSwitch: Nearly Optimal Polynomial Commitments from Tensor Codes*. Cryptology ePrint Archive, Report 2025/2065. 2025. URL: <https://eprint.iacr.org/2025/2065>.
- [BK16] Alex Biryukov and Dmitry Khovratovich. “Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem”. In: *NDSS 2016*. The Internet Society, Feb. 2016. DOI: 10.14722/ndss.2016.23108.
- [BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. “NIZK from LPN and Trapdoor Hash via Correlation Intractability for Approximable Relations”. In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 738–767. DOI: 10.1007/978-3-030-56877-1_26.
- [BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. “How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios”. In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Berlin, Heidelberg, Dec. 2012, pp. 626–643. DOI: 10.1007/978-3-642-34961-4_38.
- [BR93] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.
- [BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. “Proofs of Work From Worst-Case Assumptions”. In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 789–819. DOI: 10.1007/978-3-319-96884-1_26.
- [BRW26] Benedikt Bünz, Ron D. Rothblum, and William Wang. *Flock: Fast Proving for Batch Boolean Computations*. Cryptology ePrint Archive, Report 2026/1329. 2026. URL: <https://eprint.iacr.org/2026/1329>.
- [BT24] Alexander R. Block and Pratyush Ranjan Tiwari. “On the Concrete Security of Non-interactive FRI”. In: *SCN 24, Part I*. Ed. by Clemente Galdi and Duong Hieu Phan. Vol. 14973. LNCS. Springer, Cham, Sept. 2024, pp. 275–296. DOI: 10.1007/978-3-031-71070-4_13.
- [Bac02] Adam Back. *Hashcash – A Denial of Service Counter-Measure*. <http://www.hashcash.org/papers/hashcash.pdf>. 2002.
- [CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. “Fiat-Shamir and Correlation Intractability from Strong KDM-Secure Encryption”. In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Cham, 2018, pp. 91–122. DOI: 10.1007/978-3-319-78381-9_4.
- [CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited (Preliminary Version)”. In: *30th ACM STOC*. ACM Press, May 1998, pp. 209–218. DOI: 10.1145/276698.276741.
- [CGJJZ23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. “Correlation Intractability and SNARGs from Sub-exponential DDH”. In: *CRYPTO 2023, Part IV*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14084. LNCS. Springer, Cham, Aug. 2023, pp. 635–668. DOI: 10.1007/978-3-031-38551-3_20.

- [CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. “SNARGs for $\mathcal{P}$ from LWE”. In: *62nd FOCS*. IEEE Computer Society Press, Feb. 2022, pp. 68–79. DOI: 10.1109/FOCS52979.2021.00016.
- [CP18] Bram Cohen and Krzysztof Pietrzak. “Simple Proofs of Sequential Work”. In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Cham, 2018, pp. 451–467. DOI: 10.1007/978-3-319-78375-8_15.
- [CPV24] Oana Ciobotaru, Maxim Peter, and Vesselin Velichkov. *The Last Challenge Attack: Exploiting a Vulnerable Implementation of the Fiat-Shamir Transform in a KZG-based SNARK*. Cryptology ePrint Archive, Report 2024/398. 2024. URL: <https://eprint.iacr.org/2024/398>.
- [CT10] Alessandro Chiesa and Eran Tromer. “Proof-Carrying Data and Hearsay Arguments from Signature Cards”. In: *ICS 2010*. Ed. by Andrew Chi-Chih Yao. Tsinghua University Press, Jan. 2010, pp. 310–331.
- [Can+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. “Fiat-Shamir: from practice to theory”. In: *51st ACM STOC*. Ed. by Moses Charikar and Edith Cohen. ACM Press, June 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380.
- [DGN03] Cynthia Dwork, Andrew Goldberg, and Moni Naor. “On Memory-Bound Functions for Fighting Spam”. In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 426–444. DOI: 10.1007/978-3-540-45146-4_25.
- [DMWG23] Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. “Weak Fiat-Shamir Attacks on Modern Proof Systems”. In: *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2023, pp. 199–216. DOI: 10.1109/SP46215.2023.10179408.
- [DN93] Cynthia Dwork and Moni Naor. “Pricing via Processing or Combatting Junk Mail”. In: *CRYPTO’92*. Ed. by Ernest F. Brickell. Vol. 740. LNCS. Springer, Berlin, Heidelberg, Aug. 1993, pp. 139–147. DOI: 10.1007/3-540-48071-4_10.
- [DNW05] Cynthia Dwork, Moni Naor, and Hoeteck Wee. “Pebbling and Proofs of Work”. In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 37–54. DOI: 10.1007/11535218_3.
- [DP26] Benjamin E. Diamond and Jim Posen. “Polylogarithmic Proofs for Multilinears over Binary Towers”. In: *EUROCRYPT 2026, Part VII*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16547. LNCS. Springer, Cham, May 2026, pp. 3–32. DOI: 10.1007/978-3-032-25336-1_1.
- [FS87] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO’86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.
- [Fen26] Giacomo Fenzi. *How to Prove More False Statements: Fiat-Shamir Limitations on (Generated) RICS*. Cryptology ePrint Archive, Report 2026/1838. 2026. URL: <https://eprint.iacr.org/2026/1838>.
- [GK03] Shafi Goldwasser and Yael Tauman Kalai. “On the (In)security of the Fiat-Shamir Paradigm”. In: *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 102–115. DOI: 10.1109/SFCS.2003.1238185.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. “Delegating computation: interactive proofs for muggles”. In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 113–122. DOI: 10.1145/1374376.1374396.
- [GRY25] Ziyi Guan, Artur Riazanov, and Weiqiang Yuan. “Breaking Verifiable Delay Functions in the Random Oracle Model”. In: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 161–191. DOI: 10.1007/978-3-032-01907-3_6.
- [HL18] Justin Holmgren and Alex Lombardi. “Cryptographic Hashing from Strong One-Way Functions (Or: One-Way Product Functions and Their Applications)”. In: *59th FOCS*. Ed. by Mikkel Thorup. IEEE Computer Society Press, Oct. 2018, pp. 850–858. DOI: 10.1109/FOCS.2018.00085.

- [HLPT20] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. “How not to prove your election outcome”. In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2020, pp. 644–660. DOI: 10.1109/SP40000.2020.00048.
- [HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. “Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge)”. In: *53rd ACM STOC*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM Press, June 2021, pp. 750–760. DOI: 10.1145/3406325.3451116.
- [JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. “SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE”. In: *53rd ACM STOC*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM Press, June 2021, pp. 708–721. DOI: 10.1145/3406325.3451055.
- [KLV23] Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. “SNARGs and PPAD Hardness from the Decisional Diffie-Hellman Assumption”. In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 470–498. DOI: 10.1007/978-3-031-30617-4_16.
- [KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. “From Obfuscation to the Security of Fiat-Shamir for Proofs”. In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 224–251. DOI: 10.1007/978-3-319-63715-0_8.
- [KRS25] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. “How to Prove False Statements: Practical Attacks on Fiat-Shamir”. In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 3–26. DOI: 10.1007/978-3-032-01887-8_1.
- [KSS11] Jeff Kahn, Michael Saks, and Clifford Smyth. “The Dual BKR Inequality and Rudich’s Conjecture”. In: *Combinatorics, Probability and Computing* 20.2 (2011), pp. 257–266. DOI: 10.1017/S0963548310000465.
- [Kil92] Joe Kilian. “A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)”. In: *24th ACM STOC*. ACM Press, May 1992, pp. 723–732. DOI: 10.1145/129712.129782.
- [LV20] Alex Lombardi and Vinod Vaikuntanathan. “Fiat-Shamir for Repeated Squaring with Applications to PPAD-Hardness and VDFs”. In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 632–651. DOI: 10.1007/978-3-030-56877-1_22.
- [MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. “Publicly verifiable proofs of sequential work”. In: *ITCS 2013*. Ed. by Robert D. Kleinberg. ACM, Jan. 2013, pp. 373–388. DOI: 10.1145/2422436.2422479.
- [MSW20] Mohammad Mahmoody, Caleb Smith, and David J. Wu. “Can Verifiable Delay Functions Be Based on Random Oracles?” In: *ICALP 2020*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Vol. 168. LIPIcs. Schloss Dagstuhl, July 2020, 83:1–83:17. DOI: 10.4230/LIPIcs.ICALP.2020.83.
- [NHB24] Hieu Nguyen, Uyen Ho, and Alex Biryukov. *Fiat-Shamir in the Wild*. Cryptology ePrint Archive, Report 2024/1565. 2024. URL: <https://eprint.iacr.org/2024/1565>.
- [Nak08] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. <https://bitcoin.org/bitcoin.pdf>. 2008.
- [PS19] Chris Peikert and Sina Shiehian. “Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors”. In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 89–114. DOI: 10.1007/978-3-030-26948-7_4.
- [PS96] David Pointcheval and Jacques Stern. “Security Proofs for Signature Schemes”. In: *EUROCRYPT’96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 387–398. DOI: 10.1007/3-540-68339-9_33.

- [Pol22] Polygon Zero Team. *Plonky2: Fast Recursive Arguments with PLONK and FRI*. <https://github.com/0xPolygonZero/plonky2/blob/main/plonky2/plonky2.pdf>. 2022.
- [Rud88] Steven Rudich. “Limits on the Provable Consequences of One-Way Functions”. PhD thesis. University of California, Berkeley, 1988.
- [SKRBG11] Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangesamy, Colin Boyd, and Juan Manuel González Nieto. “Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols”. In: *CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. LNCS. Springer, Berlin, Heidelberg, Feb. 2011, pp. 284–301. DOI: 10.1007/978-3-642-19074-2_19.
- [Smy02] Clifford D. Smyth. “Reimer’s inequality and tardos’ conjecture”. In: *34th ACM STOC*. ACM Press, May 2002, pp. 218–221. DOI: 10.1145/509907.509942.
- [Sta21] StarkWare. *ethSTARK Documentation*. Cryptology ePrint Archive, Report 2021/582. 2021. URL: <https://eprint.iacr.org/2021/582>.
- [Tro15] John Tromp. “Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work”. In: *FC 2015 Workshops*. Ed. by Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff. Vol. 8976. LNCS. Springer, Berlin, Heidelberg, Jan. 2015, pp. 49–62. DOI: 10.1007/978-3-662-48051-9_4.
- [Val08] Paul Valiant. “Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency”. In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Berlin, Heidelberg, Mar. 2008, pp. 1–18. DOI: 10.1007/978-3-540-78524-8_1.
- [ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. “BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes”. In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 138–169. DOI: 10.1007/978-3-031-68403-6_5.