A Complexity-Theoretic Approach to Proofs of Space

Marshall Ball* Jiaxin Guan†

July 18, 2026

Abstract

A Proof of Space, PoS, as introduced by Dziembowski *et al.* [CRYPTO'15], is a two-phase protocol that enables a Prover to convince an efficient Verifier that it has allocated a large amount of persistent memory to storing some information.

To our knowledge, all existing PoS protocols are only known to be secure in the random oracle model (or under ad hoc assumptions about cryptographic assumptions). We provide an elementary framework for constructing PoS from a combination of derandomization assumptions and cryptographic assumptions.

We provide a few simple instantiations of the framework. We show that non-trivial PoS follow from (a) $\mathsf{E} = \mathsf{DTIME}[2^{\mathsf{O}(\mathsf{n})}]$ is hard for exponential-size nondeterministic circuits (an assumption introduced to show $\mathsf{AM} = \mathsf{NP}$), and (b) collision-resistant hash functions. We also show that PoS with nearly optimal parameters and interaction pattern follows from assumption (a) above and (c) SNARGs for P.

1 Introduction

A Proof of Space protocol, introduced by Dziembowski *et al.* [DFKP15], is a two-phase protocol that enables a very efficient Verifier to check that a Prover is using a lot of storage (persistently).[1] The first phase consists of a single short verifier message, but takes place over a relatively long period of time. After the first phase, the Prover is supposed to have stored some data σ of size N. In the second phase, the Verifier interacts with the Prover to check that prover has indeed committed $\approx N$ bits of random access memory to this task. It is integral that this second phase takes place over a short interval of time, else the Prover could simply expand the succinct description of σ at that moment to respond to the Verifier's queries.

A rich line of literature [DFKP15, ABFG14, RD16, ACK+16, ABP17, ABP18, AAC+17, Fis19, Rey24] has emerged following Dziembowski *et al.*'s seminal work. These works contain elegant theoretical constructions and connections to graph pebbling, however (to our knowledge) they all *critically* rely on the random oracle model to work. These proofs critically make simultaneous use of both the incompressibility and extractibility of random oracle queries, and we do not know any constructions outside of the random oracle model. In fact, we do not even know of any clean (even

*New York University. Email: marshall.ball@cs.nyu.edu. ORCID: 0000-0002-4236-3710.

†Zellic. Email: jiaxin@guan.io. ORCID: 0000-0003-1823-8845.

[1] A different definition with the same name emerged at the same time, due to Ateniese et al. [ABFG14]. This latter notion is not the subject of this paper.

non-falsifiable) assumptions about cryptographic hash functions that would suffice for the provable security of these constructions.[2]

In this work, we present a very simple approach to constructing Proofs of Space in the plain model. We provide three straightforward instantiations of our framework with various trade-offs. Before continuing, we emphasize that our focus is on elementary, pedagogically clean, generic feasibility; not sophisticated threat models and practical protocols.

1.1 This Work: An Elementary Approach to Proofs of Space

Perhaps the key difficulty in constructing Proofs of Space (PoS) without a random oracle, is that one needs a long string σ that is incompressible, and yet it should paradoxically also have a short description! The only reason this is not impossible is the time constraints on the PoS phases: the first phase is long (time T), while the second is short (time t). So, we actually have two different fine-grained notions of (in)compressibility:

1. σ can be expanded from a very short description in time T,
2. σ *cannot* be expanded from any description slightly shorter than $|\sigma|$ in time t, even when given the short description.

Thankfully, this in fact nearly corresponds to a notion in complexity theory, *computational depth* introduced by Antunes *et al.* [AFvMV06]. In essence, any PoS necessarily implies an explicit means of finding strings with a (fine-grained and computationally-sound) notion of computational depth: strings that can be highly compressed (if decompression is allowed to be inefficient) but don't admit "easy" compressions (if decompression must be fast).[3] To explain this notion, we need to first introduce some preliminaries.

Background: time-bounded Kolmogorov complexity. We require a classic notion of compressibility: the *t-time-bounded Kolmogorov complexity* of a string x, $\mathsf{K}^t(x)$, is the length of the shortest string s such that $U : s \overset{t}{\rightsquigarrow} x$, i.e. the universal Turing machine U on input s produces x in at most t steps.[4] The *conditional* t-time-bounded Kolmogorov complexity of x given y, $\mathsf{K}^t(x|y)$, is the length of the shortest string s such that $U : (s, y) \overset{t}{\rightsquigarrow} x$.

Functions of high computational depth. We are interested not just in the existence of strings with high computational depth, but also an efficient means of finding them. With this in mind, we are interested in *functions with high computational depth*.

We say a function $f : \{0,1\}^k \to \{0,1\}^N$ has (t, T) *computational depth* αN with error ε if

1. f is computable in time T,

[2] For the related but incomparable case of memory-hard functions, Ameri, Block and Blocki [ABB22] gave a construction assuming memory-hard puzzles, iO, and OWF (following a template laid out by Bitansky *et al.* [BGJ+16] for time-lock puzzles).

[3] Technically, in PoS we simply require that it is hard to find a quickly (time t) decompressible description of σ (Yao-style incompressibility), whereas computational depth says that no such description exists. Antunes et al. defined a few notions of computational depth, but in all there is no time bound T in the short description.

[4] The choice of U only matters up to an additive constant.

2. With probability at most ε over a random $x \xleftarrow{u} \{0,1\}^k$,

$$\mathsf{K}^t(f(x)|x)) \leq \alpha N.$$

In other words, $\mathsf{K}^T(f(x)) \leq |x| = k$ and yet $\mathsf{K}^t(f(x)|x) \approx N$.[5] In our setting we want k to be as small as possible, $k = N^\gamma$ for some constant $\gamma < 1$ or even $k = O(\log n)$.

In fact, in our setting of small $k = |x|$, we can observe that it suffices that $\mathsf{K}^t(f(x)) > \alpha N$ as $\mathsf{K}^t(f(x)|x) \geq \mathsf{K}^t(f(x)) - |x|$.

We will actually require a more robust notion in two respects.

Firstly, that f has high *approximate* computational depth, namely that for most x $\mathsf{K}^t(\hat{y}|x) > \alpha N$ for any y close to x in hamming distance. This is easily achieved by composing with an (explicit) list-decodable code, (E, D): $f'(x) = \mathsf{E}(f(x))$. However, our construction methodology directly gives this with essentially optimal parameters.

Secondly, that $f(x)$ remains incompressible even in the presence of *auxiliary information* (i.e. in the non-uniform setting). This roughly says that for any non-uniform time t (decompression) algorithm $\mathcal{A}$, except with negligible probability (over x) there is no short σ such that $\mathcal{A}(\sigma) = f(x)$. This robust notion is tailored to a cryptographic setting where we wish for non-uniform security. For example, an adversary may perform some expensive preprocessing to use at execution time. We note that our constructions of f immediately achieve this stronger non-uniform notion.

Having defined a notion that intuitively seems to capture a necessary element of an explicit PoS, the natural next question is how do we construct this function of high computational depth?

We observe that such a function (with high stretch) can be easily lifted from classic derandomization techniques.

Background: PRGs for nondeterministic distinguishers. The key ingredient in our construction is the notion of a *PRG against nondeterministic distinguishers* [KvMS12, SU05], $G : \{0,1\}^k \to \{0,1\}^n$. We say that G is such a seed-extending PRG against nondeterministic $t(n) = \mathsf{poly}(n)$ with error ε, if it is computable in time $T = t^{O(1)} = \mathsf{poly}(n)$ time, and fools any nondeterministic $t(n)$-time distinguisher, D:

$$\left| \Pr_{x \leftarrow \{0,1\}^k}[D(G(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^N}[D(y) = 1] \right| \leq \varepsilon.$$

Such PRGs are known to follow from assumptions that effectively say: "There are uniform computations for which nondeterminism (and nonuniformity) does not give arbitrarily large speed-ups."

In particular, relatively standard (worst-case) derandomization assumptions introduced in the context of showing $\mathsf{NP} = \mathsf{AM}$ [KvMS12, SU05]: $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ requires exponential size nondeterministic circuits. This assumption implies PRGs with exponential stretch, but just inverse polynomial (in output length) error.[6]

[5] A straight-forward fine-grained extension of Antunes et al.'s definition of *computational depth* of a string x is the difference between the length of its t-time compression and its T-time compression:$\mathsf{CD}^{t,T}(x) = \mathsf{K}^t(x) - \mathsf{K}^T(x)$.

[6] There is a black box barrier to getting negligible error PRGs from such assumptions [AASY15].

Computational depth via nondeterministic hardness. It is not difficult to see that such a seed-extending PRG G against nondeterministic hardness must have high computational depth:

1. A standard counting argument shows that a random string, $r \leftarrow \{0,1\}^N$ given a random string $s \leftarrow \{0,1\}^k$ has nearly N conditional K^t complexity with overwhelming probability.

   In particular, there are 2^N strings of length N and only 2^ℓ descriptions of length ℓ. Hence, a random r cannot be compressed to even $N - \log^2(N)$ bits, except with negligible probability.

2. Suppose (for the sake of contradiction) that $\mathsf{K}^t(G(x)) < \alpha N$ with significant probability. That means that there is a short description of $G(x)$, σ, that can expand to $G(x)$ in time t.

3. These two facts yield a nondeterministic time t distinguisher: nondeterministically guess a short description σ and check that σ expands to $y = G(x)$. If y was random, no such σ exists with very high probability.

Computational depth with overwhelming probability? Unfortunately, the noticeable error of the PRGs described above fails to achieve high computational depth with overwhelming probability. Discouragingly, there are even black-box barriers to achieving negligible error in such PRGs [AASY15].

To remedy this, we instead turn to the more nuanced notion of a *multiplicative PRG against nondeterministic distinguishers* [AIKS16, Sha25, DS26]. This notion, while failing to achieve negligible distinguishing distance, nonetheless preserves the probability of certain negligible probability events. In particular, an ε-multiplicative PRG against nondeterministic time t has the guarantee that for any nondeterministic time t distinguisher D,

$$\Pr_{x \leftarrow \{0,1\}^k}[D(G(x)) = 1] < 2 \Pr_{y \leftarrow \{0,1\}^N}[D(y) = 1] + \varepsilon.$$

Such PRGs with $\varepsilon = \mathsf{negl}(N)$ follow from the same assumptions described above [Sha25, DS26]. Moreover, our simple argument that PRG against nondeterministic distinguishers have computational depth hinged around a particular nondeterministic distinguisher, D (the compression tester), such that $\Pr_y[D(y) = 1] = \mathsf{negl}(N)$. It follows that $Pr_x[D(G(x)) = 1] = \mathsf{negl}(N)$ and hence such G has nearly maximal computational depth with overwhelming probability.

Theorem 1 (*Informal:* Explicit Incompressibility from Nondeterministic Hardness)**.** *Assume* $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ *is hard for exponential-size circuits. Then for every polynomial time* t *and every small constant approximation factor* $\rho > 0$*, there is an explicit function*

$$f : \{0,1\}^{\mathsf{poly}\log N} \to \{0,1\}^N$$

computable in polynomial time and is time t*-incompressible to even* $S = N - O(\rho N \log^2(1/\rho) - \mathsf{polylog}N)$ *bits within relative distance* ρ*. Namely, for any* t *decompression procedure,* $D : \{0,1\}^S \to \{0,1\}^N$*, except with negligible probability over random* x*,* $f(x) = y$ *is* ρ *far from any string in* $D(\{0,1\}^S)$*.*

At last, we can turn to constructing PoS from such functions with computational depth.

PoS from functions of high computational depth. We next show an elementary recipe for constructing a Proof of Space from any function f of high computational depth:

- **Initialization phase.**
  1. Verifier samples random $x \leftarrow \{0,1\}^k$, and the key to a hash function h, and sends both to Prover.
  2. Prover computes $y = f(x)$ and stores to memory. (This phase takes roughly time T, the time to evaluate f.)

     The Prover additionally computes and stores two other items at this time:

     (a) $r = h^*(y)$, the root of the Merkle tree of y using h.

     (b) π, a SNARG, or preprocessing for a succinct interactive argument for the statement "$r = h^*(f(x))$."

- **Verification phase.**
  1. Verifier samples a set of random challenge indices $i_1, \ldots, i_\ell \in [N]$ and sends them to the Prover.
  2. Prover responds with the root r and Merkle openings of $y_{i_1}, \ldots, y_{i_\ell}$, as well as a succinct proof the root of the Merkle tree r is the Merkle root of $y = f(x)$. If the prover does not respond within time $t' < t$ (where t is such that $f(x)$ cannot be compressed relative to time t), the verifier rejects.

Intuitively, the succinct argument binds the Prover to y and thus the Verifier can be assured that the openings are correct. There are some subtleties in soundness which we will address later in this overview.

We note that this naive recipe is only possible because f is an explicit function (and does not rely on the random oracle)!

Because the SNARG forces comparison to the verifier chosen $y = f(x)$, the Merkle tree only requires 2nd pre-image resistance and hence can be instantiated from one-way functions [AR23]. (However, we do not know appropriate succinct arguments from such assumptions.) Additionally, the succinct argument requires a particular structure which is not shared by all doubly efficient proof systems: the prover can effectively preprocess the whole proving strategy into a succinct state, which can be quickly used when interacting with the verifier. While we do not know how to adapt certain recursive proof systems to this framework, such as [RRR16], this is a feature of Kilian's classic PCP-based argument and any SNARG (as well as [GKR08], but this imposes undesirable structural limitations).

Warm-up: constant soundness. To prove the soundness of this framework, we wish to show that if there exists a cheating prover strategy that maintains a small state and yet can convince the verifier to accept with non-negligible probability, then f must not have had high computational depth. In other words, given a cheating prover $\mathcal{A}$ that, on challenge x, produces a small state σ_x that convinces the verifier to accept, we want to show how to extract $f(x)$ from the short description σ_x. However, the key challenge, for this to be a contradiction, our extractor must run in time t.

To warm up let's run through how to achieve *constant soundness*, a special case where the runtime restriction is not an issue. A standard Markov argument shows that a constant fraction of

x must be "good": on input x the cheating prover will (with constant probability) produce a small state σ_x that enables it to convince the verifier with constant probability, ε. Moreover, because the argument system and the Merkle tree is negligibly sound (and the cheating prover is polynomial time overall), most PoS soundness violations cannot be due to proofs of false statements. In other words, for almost all "good" x with overwhelming probability, any polynomial number proofs $\pi_1, \ldots, \pi_m$ produced by the cheating prover will be valid. Hence, if we query the prover with state σ_x on q tuples of indices $(i_1, \ldots, i_\ell)$, we can expect that nearly εq of the Prover's responses will provide valid bits of y, $(y_{i_1}, \ldots, y_{i_\ell})$, with valid openings and proofs that the root was computed correctly. Hence, by properties of the direct product (and tail bounds), we can extract nearly all the bits of $y = f(x)$ with high probability from σ_x by making $q = O(|y|/\varepsilon^2) = O(|y|)$ queries in time $O(t' \cdot |y|) < t$ (where $f(x)$ cannot be compressed relative to time t).

Note that this reduction consumes many random bits to produce an approximation of $y = f(x)$, which potentially contributes to the description of y (beyond simply σ_x and the code of the adversary). However, this randomness can be simply folded into the non-uniform advice using standard methods [Adl78].

Negligible soundness. The approach above fails dramatically if one wants to argue the protocol satisfies negligible soundness. In this case, a violating adversary only convinces the verifier with N^{-c} probability for some arbitrarily large constant c. Thus, to expect to see the adversary work correctly even just once (when making random queries) would require $q = N^c$ queries. This means that our reduction would run in time at least n^c. To reach a contradiction, this would require t, such that $f(x)$ cannot be expanded from a short description in time t, to be larger than any polynomial N^c. But the T, the time to compute f, would also have to be super-polynomial and the scheme could not be efficient overall.

Instead, we avoid making random queries and instead take a different approach. We start as before, given a cheating prover $\mathcal{A}$ that is convincing with inverse polynomial probability, "good" instances x, which admit descriptions σ_x that enable $\mathcal{A}$ to convince the verifier with probability N^{-c}, must be inverse polynomially dense. So it suffices to focus on compressing "good" x. For simplicity in this introduction, let us assume $\mathcal{A}$ is deterministic. This means the only randomness in the reduction above is the choices of index tuples made to $\mathcal{A}$. We observe that for "good" x, we do not need to simply blindly sample index bundles at random, but can instead use additional bits to encode "useful" queries where $\mathcal{A}$ is guaranteed to respond.

Our idea is (roughly) to apply the following sequence of observations:

1. If we sample randomly from the set of "useful" index bundles/tuples in $[N]^\ell$, we will recover nearly all the bits of y due to the properties of the direct product (i.e. parallel repetition) with just $q = \Theta(N/\ell)$ queries (with overwhelming probability). If told such a sequence of bundles, the reduction technique above will be very time-efficient.

2. If we can quickly encode such a sequence of $q = O(N/\ell)$ bundles of ℓ indices in $[N]$ using $m \ll \ell$ bits to per bundle, then overall we are only using $m \cdot q \ll \ell \cdot N/\ell = N$ additional bits to describe an approximation of y. Thus, if we can do this, the reduction will be efficient in its use of additional description length (in addition to σ_x), as well as being efficient in time.

3. Because useful tuples are N^{-c}-dense in the set of all queries, we can effectively apply a *coding theorem* to encode these tuples using just $O(\log N)$ bits. In particular, with high probability over a random $\ell \log(N) \times O(\log N)$ dimensional matrix M, for many useful tuples

$(i_1, \ldots, i_\ell) = Ms$ for some $s \in \{0,1\}^{O(\log N)}$. Thus, so long as we set $\ell \ggg \log N$, we will arrive at a contradiction.

We do not exactly implement this approach as described. First, while we observe a random matrix suffices[7] to instantiate the approach, we actually replace the linear map above with an explicit ε-Hitting Set Generator (HSG) [AIKS16]. This HSG follows from the same assumption as the incompressible function and has a constant size description. Additionally, we use this HSG to hit an entire sequence of "useful," "covering" tuples at once (as well as any random bits required for a randomized $\mathcal{A}$ to be successful on this sequence). Because the event that all the tuples are useful and cover most of y occurs with probability significantly greater than 2^{-N} we can obtain a short compression of y.

Instantiating the framework. We consider two ways of instantiating the succinct argument, with various tradeoffs. (Recall that in addition to a succinct argument, both instantiations additionally rely on the derandomization-style nondeterministic hardness assumption: E is hard for exponential-size nondeterministic circuits)

1. Perhaps the most straightforward way, is using a SNARG for P [CJJ22, HJKS22]. This way very little additional storage overhead is required to store the proof. On the other hand, SNARGs for P is a relatively strong assumption.

2. On the other end of the spectrum, observe that if the argument is implemented with a nonadaptive version of Kilian's argument system then we only need to assume Collision-Resistant Hashing. However, in this case the prover must store a PCP of the computation used to generate y in order to later argue the Merkle root was computed correctly. The length of this PCP is $\tilde{\Omega}(T) = \tilde{\Omega}(|y|)$, which means the honest prover requires much more space than what soundness guarantees in an adversarial prover: $\alpha|y|$ space.

Theorem 2 (*Informal:* PoS via SNARGs for P)**.** *Assume E is hard for exponential-size nondeterministic circuits, one-way functions, and a semi-universal SNARG for P.*

Then, there is a Proof of Space where the honest prover stores N bits and secure against any cheating prover using $(1 - o(1))N$ storage.

Theorem 3 (*Informal:* PoS via CRHF)**.** *Assume E is hard for exponential-size nondeterministic circuits and collision-resistant hash functions.*

Then, there is a constant c such that there is a Proof of Space where the honest prover stores N bits and secure against any cheating prover using $N^{1/c}$ storage.

Going forward. We note that Goldwasser, Kalai, and Rothblum's classic proof system [GKR08] additionally allows the prover to effectively condense a proving strategy into a reasonably succinct state in a preprocessing phase (indeed this exact feature has been used in the context of recent work on nearly optimal derandomization [CT22]). Moreover, this proof system is unconditional and requires no additional assumptions, unlike those presented above. On the other hand, GKR requires interaction proportional to the depth of the computation. This means that if we instantiate our framework with GKR, the function f must be computable by a low-depth circuit. This is somewhat diametrically opposed to the 2nd phase timing constraint, and computational depth. So, while this

[7] Actually, we prefer to use a union of such linear maps, as it yields a more compressed description.

would yield a PoS where the only "cryptographic" assumption was one-way functions, the resulting proof system would only be sound against non-parallel provers (who would be forced to sequentially evaluate the function with computational depth, but low circuit depth). Additionally, it is not clear how to construct such a function from the generic derandomization assumptions described above and one would need a more fine-grained tailor made assumption (similar to those used in optimal derandomization).

Whether other doubly-efficient proof systems can be adapted to this setting is an important question that could potentially immediately reduce the assumption footprint.

A second problem, which we do not address here, is composition. In particular, imagine a single cheating prover is interacting with many verifiers. It is desirable that the prover should be required to allocate storage that scales directly in proportion with number of such interactions, i.e. the prover should *not* be able to amortize the storage costs. To achieve this using our framework it suffices to construct a function f whose computational depth does not amortize (k instances are k-times as incompressible).

Our approach here immediately yields this for small k simply because incompressibility follows from f being a PRG and the direct product of PRGs remains pseudorandom via a hybrid argument. However, because these are not cryptographic PRGs, we can only tolerate distinguishers of bounded size and hence only so many hybrids. Giving a construction that remains incompressible for any polynomial number of parallel instances would yield a PoS which much more desirable security properties.

Finally, while prior pebbling-based ROM approaches to PoS have tended to additionally yield related objects such as Memory-Hard Functions, it is not immediately clear how to adapt the ideas here to get explicit constructions of such objects from relatively simple complexity-theoretic assumptions. New ideas are needed to address this important open question.

2 Preliminaries

We use $(\mathsf{out}_P, \mathsf{out}_V) \leftarrow \langle P(\mathsf{in}_P), V(\mathsf{in}_V)\rangle(\mathsf{in})$ to denote an interactive protocol between two parties P and V on a shared input in, local inputs in_P and in_V, with local outputs $\mathsf{out}_P, \mathsf{out}_V$.

Definition 1 (Proofs of Space [DFKP15])**.** Let $\mathsf{params} = (\lambda, N)$ be the security and space parameters. A Proof of Space (PoS) is an interactive protocol between two parties $P = (P_1, P_2)$ and $V = (V_1, V_2)$, with the following two phases.

- **Initialization**: $(S, \Phi) \leftarrow \langle P_1, V_1\rangle(\mathsf{params})$. The prover P_1 produces a storage advice S of size N, and the verifier V_1 may output a special symbol $\Phi = \perp$ to abort when interacting with a cheating prover.
- **Execution**: $(\varnothing, 1/0) \leftarrow \langle P_2(y), V_2(\Phi)\rangle(\mathsf{params})$. The verifier outputs a single bit $1/0$ representing either accept or reject.

We require the following properties for a Proof of space.

- **Completeness**: For any honest prover P,

$$\Pr\left[\mathsf{out} = 1 : \begin{array}{c}(S, \Phi) \leftarrow \langle P_1, V_1\rangle(\mathsf{params}),\\ (\varnothing, \mathsf{out}) \leftarrow \langle P_2(S), V_2(\Phi)\rangle(\mathsf{params})\end{array}\right] = 1$$

- (s, t, ε)-**Soundness**: For all adversarial provers $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with the size of the advice $|S| \leq s$, and $\mathcal{A}_2$ using at most t time, we have

$$\Pr\left[\mathsf{out} = 1 : \begin{array}{c}(S, \Phi) \leftarrow \langle \mathcal{A}_1, V_1 \rangle(\mathsf{params}), \\ (\varnothing, \mathsf{out}) \leftarrow \langle \mathcal{A}_2(S), V_2(\Phi) \rangle(\mathsf{params})\end{array}\right] \leq \varepsilon.$$

- **Efficiency**: The prover P_2 and the verifiers V_1, V_2 should be efficient, more specifically, should run in time $\mathsf{polylog}(N)$ and $\mathsf{poly}(\lambda)$. P_1 for the Initialization phase should run in time $\mathsf{poly}(N)$.

2.1 Vector Commitment through Merkle Tree

One building block we will be using for both our constructions is a Vector Commitment scheme instantiated through a Merkle Tree hash. Specifically, we will follow the syntax by [AR23]. We reproduce the definition below, with slight adaptations to present the scheme as a tuple of algorithms $(\mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$.

Definition 2 (Merkle Tree)**.** Fix $N, n_{\mathsf{in}}, n_{\mathsf{out}} \in \mathbb{N}$ and let Σ be a finite alphabet. Given a vector of inputs $\mathbf{y} = (y_1, \ldots, y_N) \in \Sigma^N$ and hash functions $h_1, \ldots, h_d : \Sigma^{n_{\mathsf{in}}} \to \Sigma^{n_{\mathsf{out}}}$ with $d = \log_{n_{\mathsf{in}}/n_{\mathsf{out}}}(N)$, the Merkle Tree $T = T(\mathbf{y}, h_1, \ldots, h_d)$ is defined as follows:

- The tree has $d + 1$ layers, indexed from 1 (root) to $d + 1$ (leaves).
- There are $L = \lceil N/n_{\mathsf{in}} \rceil$ leaf node **blocks**. The j-th leaf node block, denoted $c_{d+1,j}$, contains the node values $y_{(j-1)n_{\mathsf{in}}+1}, \ldots, y_{j \cdot n_{\mathsf{in}}}$.
- For $i \in [d]$, the i-th layer is denoted by w_i and is created by applying h_i to each block of the $(i+1)$-st layer: layer i contains $h_i(c_{i+1,j})$ for $j \in [|w_{i+1}|/n_{\mathsf{in}}]$, and is further divided into n_{in}-blocks $c_{i,j}$ for $j \in [|w_i|/n_{\mathsf{in}}]$ where $|w_i| = \frac{n_{\mathsf{out}}}{n_{\mathsf{in}}}|w_{i+1}|$.
- The root is $c = h_1(c_{2,1}) \in \Sigma^{n_{\mathsf{out}}}$.

Kindly note that the $c_{i,j}$'s are not the nodes of the Merkle Tree, but rather the blocks of the nodes, each containing n_{in} nodes. For more details, see [AR23].

Definition 3 (Valid Path)**.** Let $N, n_{\mathsf{in}}, n_{\mathsf{out}}, \Sigma, L, d$ and $h_1, \ldots, h_d$ be as in the Merkle Tree definition above. Given a root $c \in \Sigma^{n_{\mathsf{out}}}$ and a leaf block index $j \in [L]$, a path $\mathbf{p} = (p_1, \ldots, p_{d+1}, c_2, \ldots, c_{d+1})$ is called a *valid path* from leaf block j to root c if it satisfies:

- The path starts at the root: $p_1 = c$;
- $\forall i \in [d]$: $c_{i+1} \in \Sigma^{n_{\mathsf{in}}}$ and contains the block p_{i+1};
- $\forall i \in [d]$: $p_i = h_i(c_{i+1})$.

Definition 4 (Vector Commitment)**.** Let $\mathcal{H}$ be a family of hash functions. A Vector Commitment scheme from Merkle tree is a tuple of deterministic algorithms $(\mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$:

- $\mathsf{Commit}(\mathsf{params}, \mathbf{h} = (h_1, \ldots, h_d), \mathbf{y} = (y_1, \ldots, y_N)) \to (c, \mathsf{st})$: The committing algorithm takes as input the parameters params, a vector of hash functions $h_1, \ldots, h_d \in \mathcal{H} : \Sigma^{n_{\mathsf{in}}} \to \Sigma^{n_{\mathsf{out}}}$, and a vector of inputs $y_1, \ldots, y_N \in \Sigma$. It computes the Merkle Tree $T = T(\mathbf{y}, h_1, \ldots, h_d)$, and outputs the root $c \in \Sigma^{n_{\mathsf{out}}}$ along with a state $\mathsf{st} = (\mathbf{y}, h_1, \ldots, h_d)$.

- $\mathsf{Open}(\mathsf{params}, \mathsf{st}, c, i) \to \mathbf{p}$: The opening algorithm takes as input the parameters params, the state st, the commitment c, and an index $i \in [N]$. It outputs the opening $\mathbf{p} = (p_1, \ldots, p_{d+1}, c_2, \ldots, c_{d+1})$ corresponding to the Merkle path from leaf block $i' = \lceil \frac{i}{n_{\mathsf{in}}} \rceil$ to root c.
- $\mathsf{Verify}(\mathsf{params}, c, i, y_i, \mathbf{p}) \to 1/0$: The verification algorithm takes as input the parameters params, the commitment c, an index $i \in [N]$, the claimed value $y_i \in \Sigma$, and a proof $\mathbf{p}$, and outputs accept if and only if $\mathbf{p}$ is a valid Merkle path from the root c to leaf block $i' = \lceil \frac{i}{n_{\mathsf{in}}} \rceil$ and y_i is the i-th symbol of the committed vector.

Instantiating the hash function family $\mathcal{H}$ with a family of Collision Resistant Hash Functions (CRHFs) yields a stronger *position binding* property, which we need for our second construction that uses a Kilian-style protocol.

Definition 5 (Position Binding)**.** We say the Vector Commitment scheme is ε-position-binding if for a sufficiently large n_{in}, uniformly sampled hash functions $\mathbf{h} \leftarrow \mathcal{H}^d$, and for all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr\left[\mathsf{Verify}(\mathsf{params}, c, i, y^*, \mathbf{p}^*) = 1 \ \wedge \ y^* \neq y_i \middle| \begin{array}{c} \mathbf{y} \leftarrow \mathcal{A}_1(\mathsf{params}, \mathbf{h}) \\ (c, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{params}, \mathbf{h}, \mathbf{y}) \\ (y^*, \mathbf{p}^*, i) \leftarrow \mathcal{A}_2(\mathsf{params}, \mathbf{h}, \mathbf{y}, c, \mathsf{st}) \end{array}\right] < \varepsilon,$$

where probability is taken over the distribution of the hash functions and the random coins of the adversary.

Lemma 1 ([Mer88])**.** *Assuming Collision Resistant Hash Functions, there exists a position-binding Vector Commitment scheme.*

Alternatively, we can instantiate the hash family $\mathcal{H}$ with universal one-way hash functions. This would not give us full position binding, but instead, gives us 2nd pre-image resistance, which is sufficient for our first construction based on SNARGs for P.

Definition 6 (2-PLOSC [AR23])**.** We say the Vector Commitment scheme is an ε-secure Second-Preimage Locally Openable Succinct Commitment (2-PLOSC) if for a sufficiently large n_{in}, uniformly sampled hash functions $\mathbf{h} \leftarrow \mathcal{H}^d$, and for all inputs $\mathbf{y}$, and non-uniform PPT adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{params}, c, i, y^*, \mathbf{p}^*) = 1 \ \wedge \ y^* \neq y_i \middle| \begin{array}{c} (c, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{params}, \mathbf{h}, \mathbf{y}) \\ (y^*, \mathbf{p}^*, i) \leftarrow \mathcal{A}(\mathsf{params}, \mathbf{h}, \mathbf{y}, c, \mathsf{st}) \end{array}\right] < \varepsilon,$$

where probability is taken over the distribution of the hash functions and the random coins of the adversary.

Lemma 2 ([AR23])**.** *Assuming Universal One-Way Hash Functions (UOWHFs), there exists a 2-PLOSC.*

And importantly, one way functions imply UOWHFs.

Lemma 3 ([Rom90, KK05])**.** *The existence of one-way functions implies the existence of universal one-way hash functions.*

2.2 Non-Interactive (Semi-)Universal Arguments

For our first construction of Proof of Space, we will use non-interactive (semi-)universal arguments.

Definition 7 (Universal Language)**.** Let $\mathcal{L}_{\mathcal{U}}$ be the language of all tuples $(M, \mathsf{cf}, \mathsf{cf}', T)$ where M is a deterministic Turing machine starting from configuration cf that ends up in configuration cf' in T steps. Here, the configuration includes the machine's states and its entire memory.

Definition 8 (Non-Interactive Universal Arguments [BPSS23])**.** An non-interactive argument for $\mathcal{L}_{\mathcal{U}}$ is a tuple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{vk})$: The key generation algorithm takes as input the security parameter λ and outputs a prover key pk, and a verifier key vk.
- $\mathsf{Prove}(\mathsf{pk}, y) \to \pi$: The prover algorithm takes as input the prover key pk and an instance $y \in \mathcal{L}_{\mathcal{U}}$, and outputs a proof π.
- $\mathsf{Verify}(\mathsf{vk}, y, \pi) \to 1/0$: The verifier algorithm takes as input the verifier key vk, an instance y, and a proof π, and outputs a single bit representing accept or reject.

We require the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$ and $y = (M, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{L}_{\mathcal{U}}$ such that $|y|, T \leq 2^\lambda$, we have
$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, y, \pi) = 1 : \begin{matrix}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{pk}, y)\end{matrix}\right] = 1.$$
- **ε-Universal Soundness:** Let $\overline{T}(\lambda) = 2^\lambda$. For all $\mathsf{poly}(\lambda)$-sized adversaries $\mathcal{A}$, we require that for all $\lambda \in \mathbb{N}$:
$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, y, \pi) = 1 \wedge y \notin \mathcal{L}_M \wedge T \leq \overline{T}(\lambda) : \begin{matrix}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ (y = (M, \mathsf{cf}, \mathsf{cf}', T), \pi) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk})\end{matrix}\right] \leq \varepsilon.$$
- **Efficiency:** We require Gen to run in time $\mathsf{poly}(\lambda)$, Prove in time $\mathsf{poly}(\lambda, |y|, T)$, and Verify in time $\mathsf{poly}(\lambda, |y|)$, and the proof π has length $\mathsf{poly}(\lambda)$.

Remark 1. We can consider non-interactive universal arguments in the Common Reference String (CRS) model, where the prover key pk and the verifier key vk can both be computed from the crs. In this setting, we will overload the syntax by omitting the Gen procedure, and having Prove and Verify take crs as input instead, i.e. $\mathsf{Prove}(\mathsf{crs}, y) \to \pi$ and $\mathsf{Verify}(\mathsf{crs}, y, \pi) \to 1/0$. The security notions remain the same, as in the Soundness experiment, the adversary $\mathcal{A}$ is allowed access to both pk and vk, which can be derived from the crs.

Definition 9 (Non-Interactive Semi-Universal Arguments)**.** We can also consider non-interactive *semi-universal* arguments, where we replace ε-Universal Soundness with ε-Semi-Universal Soundness, as defined below:

- **ε-Semi-Universal Soundness:** For every polynomial $\overline{T} = \overline{T}(\lambda)$ and $\mathsf{poly}(\lambda)$-sized adversary $\mathcal{A}$, we require that for all $\lambda \in \mathbb{N}$:
$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, y, \pi) = 1 \wedge y \notin \mathcal{L}_M \wedge T \leq \overline{T}(\lambda) : \begin{matrix}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ (y = (M, \mathsf{cf}, \mathsf{cf}', T), \pi) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk})\end{matrix}\right] \leq \varepsilon.$$

Remark 2. We can further relax the Soundness properties by giving the adversary $\mathcal{A}$ only the pk, not the vk. In this case, we achieve a weaker notion of soundness, and we say the scheme is *privately verifiable*.

2.3 Kilian's Protocol

For our second construction of Proof of Space, we will use interactive arguments, e.g. Kilian's protocol.

Kilian's protocol is based on the idea of using Probabilistically Checkable Proofs (PCPs) to argue that a committed string encodes a valid transcript for a NP (or more general) relation.

Definition 10 (Probabilistically Checkable Proof (PCP))**.** Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation. A *probabilistically checkable proof* (PCP) system for R is a pair of PPT algorithms $\mathsf{PCP} = (P, V)$ with the following syntax:

- $P(x, w) \to \Pi$: On common input x and prover input w (a witness for x), output a PCP string $\Pi = (\Pi_1, \ldots, \Pi_\ell) \in \Sigma^\ell$.
- $V(x; \rho)$: On input x and randomness $\rho \in \{0,1\}^r$, the verifier makes q oracle queries to Π at indices in a set $\mathcal{Q}(\rho) = \{j_1, \ldots, j_q\} \subseteq [\ell]$ (possibly with repetitions). We write $V^{\Pi}(x; \rho)$ for the verifier that receives oracle answers $\Pi[j]$ for each $j \in \mathcal{Q}(\rho)$.

For a query set $\mathcal{Q} \subseteq [\ell]$ and answers $\mathsf{ans} \in \Sigma^{|\mathcal{Q}|}$, we write $V^{[\mathcal{Q},\mathsf{ans}]}(x; \rho)$ for the verifier run on randomness ρ when each query $j \in \mathcal{Q}(\rho)$ is answered with $\mathsf{ans}[j]$; if $\mathcal{Q}(\rho) \not\subseteq \mathcal{Q}$, then $V^{[\mathcal{Q},\mathsf{ans}]}(x; \rho)$ outputs 0. We require:

- **Completeness:** If $(x, w) \in R$, then
$$\Pr\left[V^{\Pi}(x; \rho) = 1 : \rho \leftarrow \{0,1\}^r,\ \Pi \leftarrow P(x, w)\right] = 1.$$
- $\varepsilon_{\mathsf{pcp}}$**-Soundness:** For all x with no w such that $(x, w) \in R$, and all $\tilde{\Pi} \in \Sigma^\ell$,
$$\Pr\left[V^{\tilde{\Pi}}(x; \rho) = 1 : \rho \leftarrow \{0,1\}^r\right] \leq \varepsilon_{\mathsf{pcp}}.$$
- **Efficiency:** P runs in time $\mathsf{poly}(|x|, |w|)$; V runs in time $\mathsf{poly}(|x|, r)$ and makes q queries.

We call ℓ the *proof length*, q the *query complexity*, and r the *verifier randomness length*.

Kilian's argument [Kil92] combines a PCP with a vector commitment to the PCP string: the verifier checks PCP soundness on the queried symbols, and the commitment binds those symbols to a single proof Π.

Definition 11 (Kilian commit-and-prove protocol)**.** Let R be a relation and let $\mathsf{PCP} = (P, V)$ be a PCP for R with proof length ℓ. Let MT be a position-binding vector commitment scheme instantiated with Collision Resistant Hash Functions. The *Kilian commit-and-prove protocol* for R is the following two-phase protocol on input x:

- **Commitment (preprocessing):** The prover with witness w computes $\Pi \leftarrow P(x, w)$ and $(c, \mathsf{st}) \leftarrow \mathsf{MT.Commit}(\mathsf{params}, \Pi)$, and sends c to the verifier.
- **Argument (one round):** The verifier sends $\rho \leftarrow \{0,1\}^r$; the prover computes $\mathcal{Q}$ from $V(x; \rho)$, $\mathsf{pf} \leftarrow \mathsf{MT.Open}(\mathsf{params}, \mathsf{st}, c, \mathcal{Q})$, sets $\mathsf{ans} = \Pi[\mathcal{Q}]$, and sends $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf})$; the verifier accepts iff $\mathsf{MT.Verify}(\mathsf{params}, c, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$ and $V^{[\mathcal{Q},\mathsf{ans}]}(x; \rho) = 1$.

Definition 12 (Computational soundness of Kilian)**.** The protocol in Definition 11 has *computational soundness error* $\varepsilon_{\mathsf{kil}}$ if for all PPT provers $\mathcal{P}^*$ and all x with no witness w such that $(x, w) \in R$,

$$\Pr[\text{verifier accepts on input } x] \leq \varepsilon_{\mathsf{kil}},$$

where probability is over the verifier's randomness ρ and the coins of $\mathcal{P}^*$. If MT is ε-secure and PCP has soundness $\varepsilon_{\mathsf{pcp}}$, then $\varepsilon_{\mathsf{kil}} \leq \varepsilon_{\mathsf{pcp}} + q \cdot \varepsilon$ for query complexity $q = |\mathcal{Q}|$. With MT a position-binding vector commitment scheme and negligible $\varepsilon_{\mathsf{pcp}}$, one obtains $\varepsilon_{\mathsf{kil}} = \mathsf{negl}(\lambda)$ [Kil92].

2.4 Error Correcting Codes

Definition 13 (Error Correcting Codes)**.** An Error Correcting Code (ECC) for message space $\{0,1\}^n$ and encoding space $\{0,1\}^{p(n)}$ is a tuple of polynomial time algorithms $\mathsf{ECC} = (\mathsf{En}, \mathsf{De})$ with the following syntax:

- $\mathsf{En}(m) \to c$: The encoding procedure takes as input a message $m \in \{0,1\}^n$ and outputs its corresponding codeword $c \in \{0,1\}^{p(n)}$.
- $\mathsf{De}(c) \to m$: The decoding procedure takes a codeword $c \in \{0,1\}^{p(n)}$ and outputs the corresponding message $m \in \{0,1\}^n$.

Correctness requires that for all $m \in \{0,1\}^n$, $\mathsf{De}(\mathsf{En}(m)) = m$.

An ECC has relative distance $\alpha > 0$ if for all message lengths $n \in \mathbb{N}$ and any $m_0 \neq m_1 \in \{0,1\}^n$, we have

$$\Delta(\mathsf{En}(m_0), \mathsf{En}(m_1)) \geq \alpha p,$$

where Δ denotes the Hamming distance.

A relative distance of α means the ECC is capable of correcting up to $\frac{1}{2}\alpha p$ errors, i.e., for all $m \in \{0,1\}^n$ and $c' \in \{0,1\}^{p(n)}$, if $\Delta(\mathsf{En}(m), c') < \frac{1}{2}\alpha p$, then $\mathsf{De}(c') = m$.

It is known binary error correcting codes with relative distance $\approx \frac{1}{2}$ exists with a linear blowup in encoding length.

Lemma 4 ([GS00])**.** *For all $\epsilon > 0$, there exists an error correcting code ECC with relative distance $\frac{1}{2} - \varepsilon$ and with $p = p(n) = O_\varepsilon(n)$.*

2.5 Hitting Set Generator (HSG)

Definition 14 (Hitting Set Generator (HSG))**.** Let $\mathcal{C}$ be a class of boolean functions $C : \{0,1\}^n \to \{0,1\}$. A function $G : \{0,1\}^r \to \{0,1\}^n$ is an ε-HSG for $\mathcal{C}$ if for every $C \in \mathcal{C}$ such that $\Pr[C(U_n) = 1] > \varepsilon$, there exists $x \in \{0,1\}^r$ such that $C(G(x)) = 1$.

Lemma 5 (Explicit HSG [AIKS16])**.** *If $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ is hard for exponential size nondeterministic circuits, then for every constant $b > 1$, there exists a constant $c > 1$ such that for every sufficiently large n, there is a function $G : \{0,1\}^r \to \{0,1\}^n$ that is an ε-HSG for size n^b circuits, with $r = \log(1/\varepsilon) + c \log n$. Furthermore, G is computable in time $\mathsf{poly}(n^b)$.*

Lemma 6 (Monte Carlo HSG)**.** *There is a family $\mathcal{G} = \{G : [r] \times (\{0,1\}^d \setminus \{0^n\}) \to \{0,1\}^n\}$ such that any $G \leftarrow \mathcal{G}$ can be sampled with $r(2n+d-1)$ random bits and computed in time $O(rnd)$ and for any set S with density ϵ,*

$$\Pr_{G \leftarrow \mathcal{G}}[\forall x \in [r] \times (\{0,1\}^d \setminus \{0^n\}), G(x) \notin S] \leq \left(\frac{1}{(2^d-1)\epsilon}\right)^r.$$

Hence, for any $\delta \in (0,1)$ and $\mathcal{S} = \{S_1, \ldots, S_M\}$ where each $S_i \in \mathcal{S}$ has density $\geq \epsilon$, if we take $d \geq \log(1/\epsilon) + 2$ and $r = \log(M) + \log(1/\delta)$ then

$$\Pr_{G \leftarrow \mathcal{G}}[\exists S \in \mathcal{S}, \forall x, G(x) \notin S] \leq \delta.$$

Sketch. Take G to be the union of r linear maps defined by Toeplitz matrices. The first coordinate in $[r]$ defines which linear map to apply the second input to. The Toeplitz family is pairwise independent (provided we exclude 0^n), hence Chebyshev bound will give that a single random map fails to hit any particular S with probability at most $p = 1/(2^d-1)\epsilon$. The probability of this event under r independent trials is at most p^r. A specific choice of parameters and union bound over $\mathcal{S}$ yields the last line. □

3 Hard Functions with High Time-Bounded Kolmogorov Complexity

Our constructions will rely on a central tool, a function $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ that can be computed in time T, but has a high conditional time-bounded Kolmogorov complexity $\mathsf{K}^t(f(x) \mid x)$ for time $t \ll T$. Specifically, we will consider three different variants of time-bounded Kolmogorov complexity.

We first start with the standard definition of time-bounded Kolmogorov complexity. Towards the definition, we will use the universal Turing machine U that can emulate any Turing machine with polynomial overhead. The universal Turing machine U takes as input (the description of) a program $\Pi = (M, w) \in \{0,1\}^*$, where M is a Turing machine and $w \in \{0,1\}^*$ is an input to M. We use $U(M, w, 1^t)$ to denote the output of $M(w)$ in t steps when emulated on U.

Definition 15 (Time-Bounded Kolmogorov Complexity)**.** Let U be a universal Turing machine. The *t-time-bounded Kolmogorov complexity* of s string y conditioned on x, denoted $\mathsf{K}^t(y \mid x)$, is defined as

$$\mathsf{K}^t(y \mid x) = \min_{M \in \{0,1\}^*} \left\{ |M| : U(M, x, 1^t) = y \right\}.$$

That is, M is a program such that emulating M on input x outputs y in t steps.

We say $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ has *t-time Kolmogorov complexity at least s* if

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\mathsf{K}^t(f(x) \mid x) > s \right] > 1 - \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings[8].

[8]Here we assume the input distribution for f is the uniform distribution. One could also define the notion analogously for arbitrary input distributions.

Next, we strengthen the notion to account for probabilistic Turing machines. We will let U be a universal Turing machine that emulates *probabilistic* Turing machines with polynomial overhead. The universal Turing machine U takes as input (the description of) a program $\Pi = (M, w) \in \{0,1\}^*$, where M is a Turing machine and $w \in \{0,1\}^*$ is an input to M, as well as a randomness $r \in \{0,1\}^*$. We use $U(M, w, r, 1^t)$ to denote the output of $M(w; r)$ in t steps when emulated on U, where r is fed to M as a random tape.

Definition 16 (Time-Bounded Probablistic Kolmogorov Complexity [GKLO22])**.** Let U be a universal Turing machine that emulates probabilistic Turing machines. The *t-time-bounded probabilistic Kolmogorov complexity* of a string y conditioned on x, denoted $\mathsf{pK}^t(y \mid x)$, is defined as

$$\mathsf{pK}^t(y \mid x) = \min \left\{ k \in \mathbb{N} \;\middle|\; \Pr_{r \leftarrow \mathcal{U}_t} \left[\exists M \in \{0,1\}^k : U(M, x, r, 1^t) = y \right] \geq \frac{2}{3} \right\},$$

where $\mathcal{U}_t$ denotes the uniform distribution over t-bit strings.

We say $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ has *t-time probabilistic Kolmogorov complexity at least s* if

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\mathsf{pK}^t(f(x) \mid x) > s \right] > 1 - \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings.

One way to view the definition above is that the randomness r is provided to the Turing machine M as uniform advice. Naturally, one can consider the stronger notion where *an arbitrary non-uniform advice* is provided to the Turing machine M. However, such a notion no longer makes sense to measure specific strings (one can simply write the string in the advice). Though it does make sense as a measure of the incompressibility of a distribution over strings.

Definition 17 (Kolmogorov Complexity with non-uniform Advice)**.** Let U be a universal Turing machine. The *t-time-bounded Kolmogorov complexity with non-uniform advice z* of a string y conditioned on x, denoted $\mathsf{aK}^t_z(y \mid x)$, is defined as

$$\mathsf{aK}^t_z(y \mid x) = \min_{M \in \{0,1\}^*} \left\{ |M| : U(M, (x, z), 1^t) = y \right\}.$$

That is, M is a program that outputs y given input x and a non-uniform advice z in time t. Notice that trivially, $\mathsf{aK}^t_y(y \mid x) = O(1)$, when the non-uniform advice happens to be the output string y itself (or more generally, when the non-uniform advice can depend on the output string y).

Definition 18 (t-time Incompressibility)**.** We say a function $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ is (t, s)-*incompressible* if for any non-uniform advice $z \in \{0,1\}^*$, we have

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\mathsf{aK}^t_z(f(x) \mid x) > s \right] > 1 - \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings.

Notice the order of the quantifiers in the definition above. The non-uniform advice z cannot depend on the input x, otherwise the definition is trivial as pointed out in Definition 17. Instead, a "useful" advice z needs to work for all inputs x.

3.1 Approximate Incompressibility

To utilize these hard functions with high K^t complexity, we compose a function of certain K^t complexity with an *Error Correcting Code* to obtain a function with high *Approximate* K^t complexity.

Definition 19 (Approximate Kolmogorov Complexity)**.** Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ be a function. We say f has δ-approximate t-time Kolmogorov complexity s if

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\exists y' : \Delta(y', f(x)) < \delta \cdot |f(x)| \ \wedge \ \mathsf{K}^t(y' \mid x) \leq s \right] < \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings.

That is, with overwhelming probability over choice of x, we have $\mathsf{K}^t(y' \mid x) > s$ for all y' such that the Hamming distance $\Delta(y', f(x)) < \delta \cdot |f(x)|$. We denote this as

$$\mathsf{K}^t_{\approx\delta}(f(x) \mid x) > s.$$

Analogously, we can define approximate pK^t and (t,s)-incompressibility.

Definition 20 (Approximate Probabilistic Kolmogorov Complexity)**.** Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ be a function. We say f has δ-approximate t-time probabilistic Kolmogorov complexity s if

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\exists y' : \Delta(y', f(x)) < \delta \cdot |f(x)| \ \wedge \ \mathsf{pK}^t(y' \mid x) \leq s \right] < \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings.

That is, with overwhelming probability over choice of x, we have $\mathsf{pK}^t(y' \mid x) > s$ for all y' such that the Hamming distance $\Delta(y', f(x)) < \delta \cdot |f(x)|$. We denote this as

$$\mathsf{pK}^t_{\approx\delta}(f(x) \mid x) > s.$$

Definition 21 (Approximate t-time Incompressibility)**.** Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ be a function. We say f is *δ-approximate (t,s)-incompressible* if for any non-uniform advice $z \in \{0,1\}^*$, we have

$$\Pr_{x \leftarrow \mathcal{U}_{\ell_{\mathsf{in}}}} \left[\exists y' : \Delta(y', f(x)) < \delta \cdot |f(x)| \ \wedge \ \mathsf{aK}^t_z(y' \mid x) \leq s \right] < \varepsilon,$$

for a negligible $\varepsilon = \mathsf{negl}(\ell_{\mathsf{out}}) > 0$, where $\mathcal{U}_{\ell_{\mathsf{in}}}$ denotes the uniform distribution over ℓ_{in}-bit strings.

That is, for any non-uniform advice z, with overwhelming probability over choice of x, we have

$$\mathsf{aK}^t_z(y' \mid x) > s$$

for all y' such that the Hamming distance $\Delta(y', f(x)) < \delta \cdot |f(x)|$.

3.2 Approximate Incompressibility via List-Decoding

We observe that approximate incompressibility follows generically by composing with an efficient list-decodable code. We actually do not rely on this result and instead use a direct instantiation that is more efficient.

Theorem 4 ([GR07]). *For every field $\mathbb{F}_q$, reals $\delta \in (0,1), r \in (0, 1-H_q(\delta)], \epsilon > 0$ and integer $s \geq 1$, there exists linear codes C over $\mathbb{F}_q$ of block length n that can be list-decoded from $\delta - \epsilon$-relative errors with list size $L(n) = (n/\epsilon^2)^{O(s\epsilon^{-3}\delta/(H_q^{-1}(1-r)-\delta))}$ and rate $R = r - \frac{r}{s}\sum_{i=0}^{s-1} \frac{\delta}{H_q^{-1}(1-r+ri/s)}$. Moreover, C can be constructed in time $(n/\epsilon^2)^{O(s/(\epsilon^6 r\delta))}$ and list-decoded in time polynomial in n.*

As noted by Guruswami and Rudra [GR07], this meets the Blokh–Zyablov bound, $R_{BZ}(\rho)$ where ρ is the relative distance of the code, as $s \to \infty$:

Corollary 1 ([GR07]). *For every $0 < \rho < \frac{1}{2}$ and every $0 < \eta < \rho$, there exists an explicit family of binary linear codes that is list-decodable up to $\rho - \eta$ relative distance with list size $n^{O_{\rho,\eta}(1)}$ and rate at least $R_{BZ}(\rho) - \eta$ where n is the block length and*

$$R_{BZ}(\rho) := 1 - H_2(\rho) - \rho \int_0^{1-H_2(\rho)} \frac{dx}{H_2^{-1}(1-x)}.$$

Moreover encoding and list-decoding run in $n^{O_{\rho,\eta}(1)}$ time.

We rephrase[9] this in the following convenient form (explicit here means that both encoding and list-decoding run in time polynomial in the block length):

Corollary 2 (Simple High-rate Formulation of [GR07]). *There exists an absolute constant $c > 0$ such that for every sufficiently small $0 < \rho < \frac{1}{4}$ and every $0 < \eta < \rho$, there is an explicit family of linear binary codes with list size $n^{O_{\rho,\eta}(1)}$ and rate at least*

$$1 - \rho(\log^2(\frac{1}{\rho}) + 1/2) - \eta$$

that is explicitly list-decodable up to relative distance $\rho - \eta$.

[9] $$1 - R_{BZ}(\rho) = H_2(\rho) + \rho \int_0^{1-H_2(\rho)} \frac{dx}{H_2^{-1}(1-x)}$$

If we define $u = H_2^{-1}(1-x)$ for the purposes of substitution, then $x = 1 - H_2(u)$ and

$$dx = -H_2'(u)du = -\log_2 \frac{1-u}{u} du.$$

Additionally, note that as x goes from 0 to $1 - H_2(\rho)$, u goes from $\frac{1}{2}$ to ρ. Hence,

$$\int_0^{1-H_2(\rho)} \frac{dx}{H_2^{-1}(1-x)} = \int_\rho^{1/2} \frac{\log((1-u)/u)}{u} du.$$

For $0 < u \leq 1/2$, $\log \frac{1-u}{u} \leq \log \frac{1}{u}$. So for $\rho \in (0, 1/2)$,

$$\int_\rho^{1/2} \frac{\log((1-u)/u)}{u} du \leq \int_\rho^{1/2} \frac{\log(1/u)}{u} du = 1/2(\log^2(1/\rho) - \log^2(2)) = \log^2(1/\rho) - 1/2.$$

Additionally, $H_2(\rho) = \rho\log(1/\rho) + (1-\rho)\log(1/(1-\rho)) \leq \rho\log(1/\rho) + (1-\rho)\frac{\rho}{1-\rho} \leq \rho(\log(1/\rho)+1)$. And for $\rho < 1/4$, $\log(1/\rho) \leq \frac{1}{2}\log^2(1/\rho)$. So,

$$R_{BZ}(\rho) \geq 1 - \rho(\log^2(1/\rho) + 1/2).$$

Theorem 5. *Let* $\mathsf{params} = (\lambda, N, f)$ *where* λ *is the security parameter and* $N \gg \lambda$ *is the space parameter. Let* $T(\lambda, N), t(\lambda, N) = \mathsf{poly}(\lambda, N)$*, and* $\ell_{\mathsf{in}}(\lambda, N) = \mathsf{poly}(\lambda, \log N)$*, and* $\ell_{\mathsf{out}}(\lambda, N) = N/\mathsf{poly}(\lambda)$*.*

Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ *be a function computable in time* T *such that one of the following holds:*

(a) for $(1-\varepsilon)$ *fraction of* x*,* $\mathsf{K}^t(f(x)|x) > s(\lambda, N)$*;*

(b) for $(1-\varepsilon)$ *fraction of* x*,* $\mathsf{pK}^t(f(x)|x) > s(\lambda, N)$*;*

(c) f *is* $(t, s(\lambda, N))$*-incompressible (w.r.t. an overwhelming* $(1-\varepsilon)$ *fraction of* x*).*

Let $\mathsf{ECC} = (\mathsf{En}, \mathsf{De})$ *be an error correcting code list-decodable from relative distance* $\rho - \eta$ *with block length* $n = O(N)$*, list size* $\mathsf{poly}(n) = \mathsf{poly}(N)$ *and decoding time* $t_{\mathsf{De}} = \mathsf{poly}(n) = \mathsf{poly}(N)$*. Define* $f'(x)$ *to be the following function*

$$f'(x) = \mathsf{ECC.En}(f(x)).$$

Then for $\delta = \rho - \eta$*, we have, respectively,*

(a) $\mathsf{K}^{t-t_{\mathsf{De}}}_{\approx\delta}(f'(x) \mid x) > s(\lambda, N) - O(\log N)$*;*

(b) $\mathsf{pK}^{t-t_{\mathsf{De}}}_{\approx\delta}(f'(x) \mid x) > s(\lambda, N) - O(\log N)$*;*

(c) f' *is* δ*-approximate* $(t - t_{\mathsf{De}}, s(\lambda, N) - O(\log N))$*-incompressible.*

Proof. We prove the theorem for case (a), cases (b) and (c) follow analogously.

We prove case (a) via a reduction to the property of f. We show that if there exists a string y with $K^{t-t_{\mathsf{De}}}(y \mid x) \leq s(\lambda, N) - O(\log N)$ and $\Delta(y, f'(x)) < \delta \cdot |f'(x)|$, then we can use y to efficiently compute $f(x)$, contradicting the property of f.

Let c be a constant such that the list size of ECC is at most N^c (such c exists since the list size is $\mathsf{poly}(n)$ and $n = O(N)$). Notice that since $K^{t-t_{\mathsf{De}}}(y \mid x) \leq s(\lambda, N) - c \log N - O(1)$, this means there exists a small state st with $|\mathsf{st}| \leq s(\lambda, N) - c \log N - O(1)$ and an algorithm $\mathcal{A}$ such that $y = \mathcal{A}(x, \mathsf{st})$ and $\mathcal{A}$ runs in time $t - t_{\mathsf{De}}$.

We can construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to compute $f(x)$ as follows:

$\mathcal{B}(x, \mathsf{st}, i)$:

1. Compute $y = \mathcal{A}(x, \mathsf{st})$.
2. Compute $f(x) = \mathsf{ECC.De}(y, i)$ (Output the ith item in the list).

It should be clear to see that $\mathcal{B}$ runs in time t and takes as input a state of size at most $|\mathsf{st}| + |i| + O(1) = s(\lambda, N)$.

Correctness follows from correctness of list-decoding under the list-decoding radius. □

4 Computational Depth from Hardness against Nondeterminism

Definition 22. We say a function $G : \{0,1\}^k \to \{0,1\}^n$ is an (ε, δ)-multiplicative PRG for a computational class $\mathcal{C}$ if for any family of circuits $C \in \mathcal{C}$,

$$\Pr[C(G(\mathcal{U}_k)) = 1] \leq e^{\varepsilon} \Pr[C(\mathcal{U}_n) = 1] + \delta$$

where $\mathcal{U}_k$ and $\mathcal{U}_n$ denote random variables uniformly distributed over k-bit and n-bit strings, respectively.

Very recently, Dermer and Shaltiel [DS26], showed that the classic Shaltiel Umans construction [SU01] in conjunction with techniques developed in the context of randomness extraction [TV00, BGDM23, Sha25] can be adapted to give such a PRG against the class of size s nondeterministic circuits, with essentially optimal parameters.

Theorem 6 ([DS26])**.** *If* E *is hard for exponential-size nondeterministic circuits (i.e.* $\exists L \in \mathsf{E}$ *and constant* $\beta > 0$ *such that any family of nondeterministic circuits of size* $2^{\beta n}$ *fails to decide* L *on all but a finite set of input lengths), then for every sufficiently large* s *and every* $\delta \in [2^{-s}, 1/s]$ *there is a* $(1/s, \delta)$*-multiplicative PRG,*

$$G : \{0,1\}^{r=O(\log 1/\delta)} \to \{0,1\}^s$$

for size s *nondeterministic circuits that is computable in time* $\mathsf{poly}(s)$*.*

A distinguisher, D, is considered to be a γ-limited-nondeterministic T-time (non-uniform) computation if there exists a T-time (non-uniform) distinguisher V such that $D(x) = 1 \iff \exists w \in \{0,1\}^\gamma, V(x,w) = 1$. In other words, such a distinguisher can use nondeterministic witness of length up to γ. Any limited nondeterministic computation is obviously a nondeterministic computation.

Lemma 7. *Let* $S(t)$ *denote the time to simulate a machine for* t *steps. Let* $\delta > 0$*. Let* $G : \{0,1\}^k \to \{0,1\}^n$ *be an* $(\ln(2), \delta)$*-multiplicative PRG for time-*$(S(t) + O(n))$ $(n - \log(1/\delta))$*-nondeterministic (non-uniform) computations. Then, for any advice* $z \in \{0,1\}^*$

$$\Pr_{x \leftarrow \mathcal{U}_k}[\mathsf{aK}^t_z(G(x) \mid x) \leq n - \log(1/\delta) - k] \leq 3\delta$$

We note that if one starts with a *seed-extending* multiplicative PRG for nondeterminsitic circuits, then one will not suffer the loss of k in the compression bound above. A seed-extending PRG remains pseudorandom even when given the seed (i.e. $(X, G(X)) \approx \mathcal{U}$ for uniform seed X). In particular, applying Nisan Wigderson [NW88] to Shaltiel's multiplicative (low-stretch) PRG [Sha25] indeed yields such a PRG, albeit with worse seed length than the recent construction of Dermer and Shaltiel [DS26]. This latter construction [DS26] can perhaps be made seed extending (as was the case for Shaltiel and Uman's PRG [SU01] which it is based on), however we have not yet verified this.

Proof. Consider the nondeterministic distinguisher D_z that on input $y \in \{0,1\}^n$ accepts if and only if there exists a witness $(\sigma, x) \in \{0,1\}^{(n-\log(1/\delta))\times k}$ such that the output of simulating σ with advice z for t steps on input x yields y.

By our assumption about the model, simulating σ for t steps takes time $S(t)$ so the distinguisher runs in time $S(t)$.

First, note that by a counting argument it is the case that for any $m \in \mathbb{N}$ and $z \in \{0,1\}^*$, $\Pr[\exists x \in \{0,1\}^k, \mathsf{aK}^t_z(\mathcal{U}_n|x) \leq m] \leq 2^{m+k-n}$. So, $\Pr[\exists x \in \{0,1\}^k, \mathsf{aK}^t_z(\mathcal{U}_n|x) \leq n - \log(1/\delta) - k] \leq \delta$ and hence $\Pr[D_z(\mathcal{U}_n) = 1] \leq 3\delta$.

On the other hand, assume towards contradiction that there exists z such that $\Pr_x[\mathsf{aK}^t_z(G(x) \mid x) \leq n - \log(1/\delta)] > 3\delta$. Then, $\Pr[D(G(\mathcal{U}_k)) = 1] > 3\delta$. But because G is multiplicative PRG, we reach the following contradiction

$$\Pr[D_z(G(\mathcal{U}_k)) = 1] \leq 2\Pr[D_z(\mathcal{U}_n) = 1] + \delta \leq 3\delta.$$

□

We observe that the conclusion can actually be strengthened significantly by considering the nondeterministic distinguisher that accepts nearby strings.

Lemma 8. *Let $S(t)$ denote the time to simulate a machine for t steps. Let $\rho \in (0, 1/2)$ and $\delta > 0$. Let $G : \{0,1\}^k \to \{0,1\}^n$ be an $(\ln(2), \delta)$-multiplicative PRG for time-$(S(t) + O(n))$ nondeterministic (non-uniform) computations. Then, for any advice $z \in \{0,1\}^*$*

$$\Pr_{x \leftarrow \mathcal{U}_k}[\exists \hat{y} \in \{0,1\}^n : \Delta(\hat{y}, G(x) \leq \rho n \wedge \mathsf{aK}^t_z(\hat{y} \mid x) \leq n - \log(1/\delta) - k - O(1)] \leq 3\delta.$$

Proof. Consider the nondeterministic distinguisher D_z that on input $y \in \{0,1\}^n$ accepts if and only if there exists a witness $(\sigma, x) \in \{0,1\}^{(n-\log(1/\delta))\times k}$ such that the output of simulating σ with advice z for t steps on input x yields $\hat{y}$ with $\Delta(\hat{y}, y) \leq \rho n$. Next observe that any hamming ball of radius ρn has size at most $2^{H_2(\rho)n}$. So by a counting argument we have that for any z, m,

$$\Pr_{y \leftarrow \mathcal{U}_n}[D_z(y) = 1] \leq \sum_x \Pr_y[D_z(x, y) = 1] \leq 2^{k + H_2(\rho)n}.$$

The remainder of the proof proceeds similarly to the previous. □

By simply truncating the PRG from Theorem 6 and applying the lemma above, we get an explicit function with essentially maximal computational depth.

Corollary 3. *There are constants α, β, γ such that the following holds. If E is hard for exponential size nondeterministic circuits, then for any t such that $t = \mathsf{poly}(n)$, and any $\delta \in (2^{-\alpha t}, 1/\beta t)$ there is a function $f : \{0,1\}^{r=\gamma \log 1/\delta} \to \{0,1\}^n$ such that*

1. *f is computable in time $\mathsf{poly}(t)$*
2. *for any fixed z, $\Pr_{x \leftarrow \mathcal{U}_r}[\mathsf{aK}^t_z(f(x)) \mid x) \leq n - \log(1/\delta) - r] \leq 3\delta$.*

e.g. if we take $t = n^c$ and $\delta = 2^{-\sqrt{n}}$, then the resulting f is $(t, n - O(\sqrt{n})$-incompressible with error $\delta = 2^{-\sqrt{n}}$. Else if we take $\delta = 2^{-\mathsf{poly}\log(n)} = \mathsf{negl}(n)$, then the resulting f is $(t, n - O(\mathsf{polylog}))$ incompressible with $\mathsf{negl}(n)$ error.

Combining the above with Corollary 2 and Theorem 5 we get the following corollary that says effectively that for any arbitrarily small constant ϵ there is a constant δ such that one can get a polynomial time expanding function which cannot be δ-approximately compressed to size $(1 - \epsilon)n$ (relative to a polynomial decompression time).

Corollary 4. *Assume E is hard for exponential size nondeterministic circuits.*

Then there is a polynomial q such that for any polynomial decompression time t, every $\rho \in (0, 1/2)$, and any $\mu \in (2^{-q(t+m)}, 1/q(t + m))$, there is an explicit function

$$F : \{0,1\}^r \to \{0,1\}^m, \qquad r = O(\log(1/\mu)),$$

computable in time $\mathsf{poly}(t + m)$ such that for any fixed advice z

$$\Pr_{x \leftarrow \mathcal{U}_r}[\exists \hat{y} \in \{0,1\}^m : \Delta(\hat{y}, F(x)) < \rho m \wedge \mathsf{aK}^t_z(\hat{y} \mid x) \leq s_\rho(m, \mu)] \leq 3\mu,$$

where

$$s_\rho(m,\mu) = m - H_2(\rho)m - r - \log(1/\mu) - O(1).$$

In other words, F is ρ-approximate $(t, s_\rho(m,\mu)$-incompressible with error 3μ.

In particular if we take $\rho = o(1)$, then $H_2(\rho)m = O(\rho m \log(1/\rho))$.

In general, if a G is secure against γ-limited nondeterministic distinguishers for $\gamma < n - \log(1/\delta)$, then one can lower bound the conditional T-time-bounded Kolmogorov complexity of G's output (conditioned on the input) by γ.

We observe that limited nondeterministic hardness is in fact *necessary* for high-on-average K^t complexity (and hence for high incompressibility). Nondeterministic computation is traditionally studied in the context of decision problems and these definitions are not always amenable to computing many valued functions. In the next proposition, we consider a multivalued nondeterministic computation to be one that outputs a *set* of outputs. An *multivalued γ-limited nondeterministic computation in time t* N corresponds is defined by a deterministic time t computation M that takes an input and a witness of length γ: $N(x) := \{y : \exists w \in \{0,1\}^\gamma, M(x,w) = y\}$. We say such a computation fails to compute function if the output of the function is not contained in its output set. (We note that this is not the only way to define nondeterministic computation for multibit output, merely a simple and strong one that suffices in this context.)

Proposition 1. *Let $f : \{0,1\}^k \to \{0,1\}^n$ such that $\Pr[\mathsf{K}^t(f(\mathcal{U}_k)) \geq \ell] > 1 - \varepsilon$. Then, for any multivalued $(\ell - k - \omega(1))$-limited nondeterministic time t computation, N:*

$$\Pr_{x \sim \mathcal{U}_k}[f(x) \notin N(x)] > 1 - \varepsilon$$

Viewed this way, multivalued nondeterminism shows we can interpret this proposition as saying that any function with high K^t complexity outputs must remain hard for time t even when such computation is given arbitrary bounded leakage on the output.

Sketch. For the sake of contradiction, suppose N is such a limited nondeterministic machine such that

$$\Pr_x[\exists w, |w| \leq \ell - k \wedge N(x,w) = f(x)] \geq \varepsilon.$$

Then for any x such that the above event holds, $y = f(x)$ can be described via $\langle N \rangle$, x and w. Thus, $\mathsf{K}^t(y) \leq O(1) + |x| + |w| = O(1) + k + \ell - k - \omega(1) \leq \ell$. □

In summary, one gets a function whose outputs have large computational depth (even when given the input) from fairly standard derandomization circuit lower bounds against nondeterministic circuits. And moreover, a weaker form of lower bounds against such circuits are in fact necessary. We leave constructing PRGs against limited nondeterministic circuits from minimal circuit lower bounds to follow up work.

Finally we remark that such absolute incompressibility is not necessary in the context of PoS, only computational soundess is required. More precisely, in this setting, we only require that it is hard to find a compression in arbitrary polynomial time (not that no such compression exists).

5 Extraction Lemma

In this section, we present our core technical lemma, the extraction lemma, which will be a central ingredient in our constructions of Proof of Space.

We first go through some useful lemmas that will be needed for the final extraction lemma.

5.1 Useful Lemmas

To prove the extraction lemma, we will need the following two lemmas.

Lemma 9. *Let A be a randomized algorithm and let $\mathbb{X}$ denote a random variable. If $\Pr_{\mathbb{X}}[A(\mathbb{X}) = 1] > \varepsilon$, then*

$$\Pr_{\mathbb{X}}[\mathbb{X} \in \text{Good}] > \frac{\varepsilon}{2},$$

where $\text{Good} = \{x : \Pr[A(x) = 1] > \varepsilon/2\}$ is the set of inputs on which A accepts with probability greater than $\varepsilon/2$.

Proof. We have

$$\begin{aligned}\varepsilon < \Pr_{\mathbb{X}}[A(\mathbb{X}) = 1] = \Pr_{\mathbb{X}}[A(\mathbb{X}) = 1 \mid \mathbb{X} \in \text{Good}] \cdot \Pr_{\mathbb{X}}[\mathbb{X} \in \text{Good}] \\ + \Pr_{\mathbb{X}}[A(\mathbb{X}) = 1 \mid \mathbb{X} \notin \text{Good}] \cdot \Pr_{\mathbb{X}}[\mathbb{X} \notin \text{Good}].\end{aligned}$$

The first conditional probability is at most 1, and the second is at most $\varepsilon/2$ by definition of Good. Thus,

$$\varepsilon < \Pr_{\mathbb{X}}[\mathbb{X} \in \text{Good}] + \frac{\varepsilon}{2}.$$

Rearranging gives $\Pr_{\mathbb{X}}[\mathbb{X} \in \text{Good}] > \varepsilon/2$. □

Lemma 10 (Index Covering)**.** *Let $\beta = \beta(n) \in (0, 1)$ and $\varepsilon \in (0, 1)$. Let $\mathcal{G} \subseteq [n]^k$ be a collection of "good" k-tuples such that $|\mathcal{G}| \geq \varepsilon \cdot n^k$. Define*

$$\Lambda_{\beta,\varepsilon,k} := k \log \frac{1}{1-\beta} - \log \frac{1}{\varepsilon}$$

If $\Lambda_{\beta,\varepsilon,k} > 0$, then for any $\lambda \geq 1$, α independent samples from the uniform distribution over $\mathcal{G}$ cover at least $(1-\beta)n$ coordinates except with probability $2^{-\lambda}$, provided

$$\alpha \geq \frac{H_2(\beta)n + \lambda + 2}{\Lambda_{\beta,\varepsilon,k}}$$

In particular, let $\varepsilon \geq n^{-O(1)}$ and $k = \Theta(\log^2(n))$, then taking $\beta = 1/\sqrt{\log n}$ gives $\alpha = O(n \log\log n / \log^2 n)$.

Proof. Fix an arbitrary set $B \subseteq [n]$ of size $|B| = \lceil \beta n \rceil$. Let $\mathcal{S}_B \subseteq [n]^k$ be the set of k-tuples that avoid B entirely, i.e., tuples with all coordinates in $[n] \setminus B$. Then $|\mathcal{S}_B| \leq ((1-\beta)n)^k$.

Let S be a uniformly random k-tuple from $\mathcal{G}$. The probability that S avoids B is

$$\Pr[S \in \mathcal{S}_B] = \frac{|\mathcal{S}_B \cap \mathcal{G}|}{|\mathcal{G}|} \leq \frac{|\mathcal{S}_B|}{|\mathcal{G}|} \leq \frac{((1-\beta)n)^k}{\varepsilon \cdot n^k} = \frac{(1-\beta)^k}{\varepsilon} = 2^{-\Lambda_{\beta,\varepsilon,k}}.$$

Therefore if we sample α independent k-tuples from $\mathcal{G}$, the probability that all of them avoid B is at most $\left(\frac{(1-\beta)^k}{\varepsilon}\right)^\alpha$. Notice that

$$\frac{(1-\beta)^k}{\varepsilon} = 2^{\log(1/\varepsilon) - k\log(1/(1-\beta))} = 2^{-\Omega(k)}.$$

Now, the entropy bound gives (where H_2 is the binary entropy function):

$$\binom{n}{\lceil \beta n \rceil} \leq 2^{H_2(\beta)n+2}$$

Thus, by a union bound over all $\binom{n}{\lceil \beta n \rceil}$ choices of B, the probability that there exists some set of $\lceil \beta n \rceil$ indices not covered by any sample is at most

$$2^{H_2(\beta)n+2} \cdot \left(\frac{(1-\beta)^k}{\varepsilon}\right)^\alpha \leq 2^{H_2(\beta)n+2-\alpha\Lambda_{\beta,\varepsilon,k}}$$

By the assumed lower bound on α, this is at most $2^{-\lambda}$. □

5.2 Extraction Lemma

Finally, we present the extraction lemma. This roughly says that if there is a two-phase (polytime initialization that generates a small state to pass to a t-time second phase) algorithm that succeeds in violating the Proof of Space guarantees when the verifier has access to the *correct* commitment to the incompressible string $\boldsymbol{y}$, then $\boldsymbol{y}$ is approximately incompressible (relative to time decompression time t). Note that this decompression time is crucially independent of the time to compute f, T.

We frame our argument as using advice, however under the derandomization assumption if the adversarial prover $\mathcal{A}$ is uniform then this advice is dominated by the length of params. (We remark that at the expense of longer advice the derandomization assumption is unnecessary.)

Note that this critically does not say anything about how the verifier got the correct commitment. In Section 6 we show how to use certain succinct proof systems to do just this.

Lemma 11 (Extraction Lemma)**.** *Assume* $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ *is hard for exponential size nondeterministic circuits.*

Let $\mathsf{MT} = (\mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *be a 2-PLOSC scheme that is* ε_1*-secure. Fix approximation parameter* $\beta = \beta(N) \in (0, 1/2]$*, a query length* $k < N$*, and* $\varepsilon_2 \geq N^{-O(1)}$*. Let* $\gamma := \varepsilon_2 - k\varepsilon_1$ *and define*

$$\Lambda := k \log \frac{1}{1-\beta} - \log \frac{4}{\gamma}.$$

Assume $\gamma \geq N^{-O(1)}$ *and* $\Lambda > 0$ *and set*

$$\alpha \geq \lceil \frac{NH_2(\beta) + 3}{\Lambda} \rceil.$$

Let ℓ_h *be an upper bound on the number of bits used to sample* $\boldsymbol{h}$ *in the 2-PLOSC.*

Fix $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *where* $\mathcal{A}_0$ *is a non-uniform PPT initialization algorithm and* $\mathcal{A}_1$ *is a (randomized) non-uniform algorithm running in time* t*. Then, there is a fixed non-uniform advice string* z *(that is comprised of* params*,* $\mathcal{A}_1$*, and a HSG description) such that:*

If $\boldsymbol{y} \in \Sigma^N$ *and* x *such that* $f(x) = \boldsymbol{y}$,

$$\Pr\left[\forall j \in [k], \mathsf{Verify}(\mathsf{params}, c, i_j, y'_j, \boldsymbol{p}_j) = 1; \quad \begin{array}{c} \boldsymbol{h} \leftarrow \mathcal{H}_{\text{2-PLOSC}}, \\ (c, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{params}, \boldsymbol{h}, \boldsymbol{y}), \\ \sigma \leftarrow \mathcal{A}_0(\mathsf{params}, \boldsymbol{h}, \boldsymbol{y}, x, c) : \sigma \in \{0,1\}^m, \\ \boldsymbol{i} = (i_1, \ldots, i_k) \leftarrow [N]^k, \\ (y'_1, \ldots, y'_k, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_k) \leftarrow \mathcal{A}_1(\mathsf{params}, \boldsymbol{h}, \sigma, \boldsymbol{i}) \end{array}\right] \geq \varepsilon_2,$$

Then there is a string $\hat{\boldsymbol{y}}$ *satisfying* $\Delta(\boldsymbol{y}, \hat{\boldsymbol{y}}) \leq \beta N$ *such that*

$$\mathsf{aK}^{t'}_z(\hat{\boldsymbol{y}} \mid x) \leq m + O\left(\alpha \log \frac{1}{\gamma} + \log(N + t + \ell_h + m)\right)$$

where $t' = \mathsf{poly}(t\alpha + N + \ell_{\boldsymbol{h}} + m)$.

In particular, for $\varepsilon_1 = \mathsf{negl}(N)$ *and* $k = \Theta(\log^2 N)$, *if we take* $\beta = 1/\sqrt{\log N}$, *then we have* $\alpha = O(N \log\log N / \log^2 N)$. *Thus,*

$$\Delta(\boldsymbol{y}, \hat{\boldsymbol{y}}) \leq O(N/\sqrt{\log N}) \qquad \textit{and} \qquad \mathsf{aK}^{t'}_z(\hat{\boldsymbol{y}} \mid x) \leq m + O\left(\frac{N \log\log N}{\log N} + \log(t + \ell_h + m)\right)$$

Before proving we make some remarks about this lemma, whose proof is more general than the lemma suggests.

Remark 3. The derandomization assumption in this extraction lemma is almost completely superfluous, but we believe using it makes the resulting statement/proof cleaner (and we are already relying on it for incompressibility). This assumption is used solely to instantiate the HSG explicitly. In particular, because the advice string is comprised solely of params and descriptions of algorithms $\mathcal{A}$ and the HSG, this actually gives a bound on $\mathrm{K}^t(\hat{y} \mid x)$, with a (additive) $|\mathsf{params}| + O(1)$ loss over what is in the lemma.

On the other hand, at the expense of longer advice z, one can instead instantiate the HSG unconditionally using lemma 6. Because contradicting incompressibility only requires compression of a non-negligible fraction, one need not compress the witness set of any circuit $C_{x,\boldsymbol{y}}$ but only a significant fraction of them (see proof below for description of these circuits). Hence, one will get that for typical x on which $\mathcal{A}$ is successful, even better parameters than above:

$$\mathsf{aK}^{t'}_z(\hat{\boldsymbol{y}} | x) \leq m + O\left(\alpha \log(1/\gamma) + \log N\right).$$

Proof. Call an execution of $\mathcal{A}$ with commitment (root) c when given a tuple $\boldsymbol{i} = (i_1, \ldots, i_k)$ *accepting* if all k openings verify against c and *correct* if $y'_j = y_{i_j}$ for every $j \in [k]$.

Intuitively, the binding of the 2-PLOSC tells us that correct and accepting executions must occur nearly as frequently as accepting executions. In particular, if an execution accepts but is not correct then for some coordinate j it gives an accepting opening to c whose value is not y_j, which violates soundness. Hence by a union bound over the k coordinates and the ε_1-security of the 2-PLOSC, such an execution happens with probability at most $k\varepsilon_1$. Hence, the probability of a correct *and* accepting execution is at least $\gamma = \varepsilon_2 - k\varepsilon_1$.

Now, for each $\boldsymbol{h}$, let $(c_{\boldsymbol{h}}, \mathsf{st}_{\boldsymbol{h}}) = \mathsf{Commit}(\mathsf{params}, \boldsymbol{h}, \boldsymbol{y})$ and define

$$p_{\boldsymbol{h}} := \Pr_{\sigma \leftarrow \mathcal{A}_0(\mathsf{params}, \boldsymbol{h}, \boldsymbol{y}, x, c_{\boldsymbol{h}}), \boldsymbol{i}, \mathcal{A}_1} \left[\mathcal{A}_1(\mathsf{params}, \boldsymbol{h}, \sigma, \boldsymbol{i}) \text{ is correct and accepting with root } c_{\boldsymbol{h}}\right].$$

By Lemma 9, because $\mathbb{E}_{\boldsymbol{h}}[p_{\boldsymbol{h}}] \geq \gamma$,[10] it follows that if

$$\mathrm{Good}_{x,\boldsymbol{y}} := \{\boldsymbol{h} \mid p_{\boldsymbol{h}} \geq \gamma/2\},$$

then,

$$\Pr_{\boldsymbol{h}}[\boldsymbol{h} \in \mathrm{Good}] \geq \gamma/2.$$

For any $\boldsymbol{h} \in \mathrm{Good}$, by an averaging argument over $\mathcal{A}_0$ there must be some $\sigma_{\boldsymbol{h}}$ of length m such that

$$\Pr_{\boldsymbol{i}}[\mathcal{A}_1(\mathsf{params}, \boldsymbol{h}, \sigma_{\boldsymbol{h}}, \boldsymbol{i}) \text{ is correct and accepting with root } c_{\boldsymbol{h}}] \geq \gamma/2.$$

Fix some such $\sigma_{\boldsymbol{h}}$ for each $\boldsymbol{h}$. Now given this we can define

$$\mathcal{G}_{\boldsymbol{h}} := \{\boldsymbol{i} \mid \Pr[\mathcal{A}_1(\mathsf{params}, \boldsymbol{h}, \sigma_{\boldsymbol{h}}, \boldsymbol{i}) \text{ is correct and accepting with root } c_{\boldsymbol{h}}] \geq \gamma/4\}.$$

Again invoking Lemma 9, we have

$$\Pr[\boldsymbol{i} \in \mathcal{G}_{\boldsymbol{h}}] \geq \gamma/4.$$

Apply Lemma 10 with $\varepsilon = \gamma/4$ and $\lambda = 1$. By the bound on α, if independent $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha$ are sampled from $\mathcal{G}_{\boldsymbol{h}}$ then they will cover $(1-\beta)$-fraction of $[N]$ with probability at least $1/2$. Hence if the $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha$ are sampled uniformly from the entire space, $[N]^k$, then the probability all hit $\mathcal{G}_{\boldsymbol{h}}$ and cover $(1-\beta)$-fraction of $[N]$ is at least

$$\frac{1}{2}(\gamma/4)^\alpha$$

Now consider the set of strings comprised of the hash, tuples, and coins for $\mathcal{A}_1$ for each invocation,

$$\omega = (\boldsymbol{h}, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha, r_1, \ldots, r_\alpha).$$

Let $\mathrm{Great}_{x,\boldsymbol{y}}$ denote the set of pairs (σ, ω) such that $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha$ in $\mathcal{G}_{\boldsymbol{h}}$ and cover a $(1-\beta)$ fraction, and all invocations of $\mathcal{A}_1$ on the respective coins and state σ are correct and accepting. Then by the discussion above,

$$\Pr_{(\sigma,\omega)}[(\sigma,\omega) \in \mathrm{Great}] \geq 2^{-m} \cdot \underbrace{\gamma/4(\gamma/4)^{2\alpha}}_{p^* := \Pr_\omega[\exists \sigma, (\omega,\sigma) \in \mathrm{Great}]}.$$

Now consider the circuit $C_{x,\boldsymbol{y}}$ that has x and $\boldsymbol{y}$ hardwired and recognizes $\mathrm{Great}_{x,\boldsymbol{y}}$. It does the following on input (σ, ω):

1. Parse ω into $\boldsymbol{h}, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha, r_1, \ldots, r_\alpha$.
2. Compute $(c, \mathsf{st}) = \mathsf{Commit}(\mathsf{params}, \boldsymbol{h}, \boldsymbol{y})$.
3. Simulate α runs of $\mathcal{A}_1$ using x, state σ, indices $\boldsymbol{i}_j$ and randomness r_j (for $j = 1, \ldots, \alpha$), to get $\hat{\boldsymbol{y}}_{\boldsymbol{i}_1}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{i}_\alpha}, \hat{\pi}_1, \ldots, \hat{\pi}_\alpha$.
4. Check that all proof, output pairs $(\hat{\boldsymbol{y}}_{\boldsymbol{i}_j}, \hat{\pi}_j)$ verify.

[10] In other words, $\Pr_{\boldsymbol{h}}[\mathcal{A}^*(\boldsymbol{h}) = 1] \geq \gamma$ where $\mathcal{A}^*$ is the randomized procedure on the RHS above such that $\Pr[\mathcal{A}^*(\boldsymbol{h}) = 1] = p_{\boldsymbol{h}}$.

5. Check that all symbols in $\hat{\boldsymbol{y}}_{\boldsymbol{i}_1}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{i}_\alpha}$ match $\boldsymbol{y}$.
6. Check that $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_\alpha$ cover $(1-\beta)$ fraction of $[N]$.
7. Accept iff all checks pass.

Note that $C_{x,\boldsymbol{y}}$ is $\mathsf{poly}(t\alpha + N + \ell_h + m)$-size so we can hit a witness with a good Hitting Set Generator (HSG). Critically, this is independent of T, the time to compute f. Applying Lemma 5 to $C_{x,\boldsymbol{y}}$ with error $2^{-m}p^*/2$ yields a $\mathsf{poly}(t\alpha + N + \ell_h + m)$-time function G and seed s^* of length

$$m + O\left(\log(1/p^*) + \log\left(\alpha(k\log N + t) + N + \ell_{\boldsymbol{h}} + m\right)\right) = m + O\left(\alpha\log(1/\gamma) + \log(N + t + \ell_h + m)\right)$$

such that if $G(s^*) = (\sigma^*, \omega^*)$ then $C_{x,\boldsymbol{y}}(\sigma^*, \omega^*) = 1$. Hence we can use σ^*, ω^* and $\mathcal{A}_1$ to recover all but β-fraction of $\boldsymbol{y}$.

Thus, we can define a decompressor D_z with advice z that does just that. The advice z is comprised of params, description of $\mathcal{A}_1$, the HSG G's description. On input (x, s^*), D_z does the following:

1. Expand $G(s^*) = (\sigma^*, \omega^*)$ and parse $\omega^* = (\boldsymbol{h}^*, \boldsymbol{i}_1^*, \ldots, \boldsymbol{i}_\alpha^*, r_1^*, \ldots, r_\alpha^*)$.
2. For each $j \in [\alpha]$, compute $\mathcal{A}_1(\mathsf{params}, \boldsymbol{h}^*, \sigma^*, \boldsymbol{i}_j^*; r_j^*) = \hat{\boldsymbol{y}}_{\boldsymbol{i}_j^*}, \pi_j$.
3. Set $\hat{\boldsymbol{y}}$ to be consistent with $\hat{\boldsymbol{y}}_{\boldsymbol{i}_1^*}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{i}_\alpha^*}$ and 0 otherwise.
4. Return $\hat{\boldsymbol{y}}$.

Because $(\sigma^*, \omega^*) \in \mathrm{Great}_{x,\boldsymbol{y}}$ (and hence $\boldsymbol{i}_j$'s cover all but β-fraction and all $\hat{\boldsymbol{y}}_{\boldsymbol{i}_j}$'s returned by $\mathcal{A}_1$ match $\boldsymbol{y}$), we know that the $\hat{\boldsymbol{y}}$ returned is such that $\Delta(\boldsymbol{y}, \hat{\boldsymbol{y}}) \le \beta N$.

Moreover by inspection,

$$\mathsf{aK}_z^{t'}(\hat{\boldsymbol{y}}|x) \le m + O\left(\alpha\log(1/\gamma) + \log(N + t + \ell_{\boldsymbol{h}} + m)\right)$$

for $t' = \mathsf{poly}(t\alpha + N + \ell_h + m)$. □

6 Proof of Space Constructions

This section instantiates the framework in two ways, building on prior sections.

Throughout, let $n = \ell_{\mathsf{out}}$ be the output length of f. In this section we use one concrete extraction regime:

$$k = \Theta(\log^2 n), \qquad \beta(n) = \frac{1}{\sqrt{\log n}}.$$

The hidden constant in k is chosen large enough for every fixed inverse-polynomial residual success density. We assume that f is 2β-approximately (t_{K}, s)-incompressible with error $\varepsilon_{\mathsf{inc}}$. Thus a description that outputs any string within distance βn of $f(x)$ for an $\varepsilon_{\mathsf{inc}}$-dense set of inputs already contradicts the assumed incompressibility of f.

We use the following root-checking computation in both instantiations. The machine M_f takes $(\mathsf{params}, x, \mathbf{h}, c)$ as configuration data, computes $y = f(x)$, computes the Merkle root

$$(\hat{c}, \bot) = \mathsf{MT.Commit}(\mathsf{params}, h_1, \ldots, h_d, y),$$

and accepts if and only if $\hat{c} = c$. Its running time is denoted by T_f.

6.1 Construction from Non-Interactive Arguments

Our first construction uses a semi-universal non-interactive argument for the root-checking computation above. The timing of the commitment is important: the prover does not send the root in the initialization phase. The verifier first sends the random positions to be opened, and the prover then responds with the root, the openings, and the argument that the root is the honest root of $f(x)$.

Construction 1 (PoS from semi-universal arguments)**.** *Let* MT *be an* $\varepsilon_{\mathsf{mt}}$*-secure* 2*-PLOSC scheme and let* $\mathsf{ARG} = (\mathsf{Prove}, \mathsf{Verify})$ *be a semi-universal non-interactive argument with soundness error* $\varepsilon_{\mathsf{arg}}$ *for computations of time* T_f*. The protocol is as follows.*

- ***Initialization phase.***

Verifier $V_1(\mathsf{params})$*:*

1. *Sample* $x \leftarrow \{0,1\}^{\ell_{\mathsf{in}}}$ *and hash functions* $\mathbf{h} = (h_1, \ldots, h_d)$ *for* MT*.*
2. *Send* $(x, \mathbf{h})$ *to* P_1*.*
3. *Output verifier state* $\Phi = (x, \mathbf{h})$*.*

Prover $P_1(\mathsf{params})$*:*

1. *Receive* $(x, \mathbf{h})$*.*
2. *Compute* $y = f(x)$ *and* $(c, \mathsf{st}) = \mathsf{MT.Commit}(\mathsf{params}, h_1, \ldots, h_d, y)$*.*
3. *Compute*

$$\pi \leftarrow \mathsf{ARG.Prove}\big(\mathsf{crs}, (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c), \mathsf{cf}' = \mathsf{accept}, T_f)\big)\,.$$

4. *Store* $S = (c, \mathsf{st}, \pi)$*.*

- ***Execution phase.***

Verifier $V_2(\mathsf{params}, \Phi)$*:*

1. *Parse* $\Phi = (x, \mathbf{h})$*. Sample and send* $\nu = (i_1, \ldots, i_k) \leftarrow [n]^k$*.*
2. *Receive* c*,* π*, and openings* $\{(y_{i_j}, \mathbf{p}_j)\}_{j \in [k]}$*.*
3. *Check*

$$\mathsf{ARG.Verify}\big(\mathsf{crs}, (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c), \mathsf{cf}' = \mathsf{accept}, T_f), \pi\big) = 1.$$

4. *For every* $j \in [k]$*, check* $\mathsf{MT.Verify}(\mathsf{params}, c, i_j, y_{i_j}, \mathbf{p}_j) = 1$*.*
5. *Accept if and only if all checks accept and the response arrives within the execution time bound.*

Prover $P_2(\mathsf{params}, S)$*:*

1. *Parse* $S = (c, \mathsf{st}, \pi)$ *and receive* $\nu = (i_1, \ldots, i_k)$.
2. *For each* $j \in [k]$*, compute* $\mathbf{p}_j \leftarrow \mathsf{MT.Open}(\mathsf{params}, \mathsf{st}, c, i_j)$.
3. *Send* c*,* π*, and* $\{(y_{i_j}, \mathbf{p}_j)\}_{j \in [k]}$.

Theorem 7 (Soundness from semi-universal arguments)**.** *Assume the derandomization hypothesis required by Lemma 11.*

Let $k = \Theta(\log^2 n)$ *and* $\beta = 1/\sqrt{\log n}$ *be the parameters fixed above.*

Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^n$ *be computable in time* T_f *and be* $(\beta + \eta)$*-approximately* (t_{K}, s)*-incompressible with error* $\varepsilon_{\mathsf{inc}}$*. Let* MT *be* $\varepsilon_{\mathsf{mt}}$*-secure and let* ARG *have soundness* $\varepsilon_{\mathsf{arg}}$ *for the root-checking instances above.*

For every execution time bound τ *and every parameter* $\rho > 0$*, let* Δ_ρ *denote the additive storage loss guaranteed by Lemma 11, at radius* β *and query count* k*, when applied with success parameter* $\theta_\rho := \varepsilon_{\mathsf{inc}} + \rho/2$*, Merkle error* $\varepsilon_{\mathsf{mt}}$*, second-phase time* τ*, and state-size bound* $m \le s$*. Suppose this application of Lemma 11 is in its stated parameter range and has decompressor running time at most* t_{K}*.*

Then Construction 1, instantiated with k *execution queries, is complete, has efficient verification and honest proving time polynomial in* $T_f + n$*, and is*

$$(s - \Delta_\rho,\ \tau,\ \varepsilon_{\mathsf{arg}} + 2\varepsilon_{\mathsf{inc}} + \rho)\text{-}\textit{sound}.$$

Proof. Completeness follows from correctness of MT and completeness of ARG. The efficiency bounds are immediate from the construction: P_1 computes f, one Merkle commitment, and one argument for a time-T_f computation; P_2 opens k Merkle paths and sends one already-computed argument; and V_2 verifies one succinct argument and k Merkle paths.

It remains to prove soundness. Suppose, toward contradiction, that a prover $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with initialization state size at most $s - \Delta_\rho$ and execution time at most τ makes V_2 accept with probability

$$\mu > \varepsilon_{\mathsf{arg}} + 2\varepsilon_{\mathsf{inc}} + \rho.$$

Set $\theta_\rho = \varepsilon_{\mathsf{inc}} + \rho/2$. The probability is over x, $\mathbf{h}$, the verifier's sampled tuple $\nu \in [n]^k$, and the prover's randomness.

Let $\mathsf{FalseArg}$ be the event that V_2 accepts but the statement accepted by ARG is false, namely

$$\mathsf{MT.Commit}(\mathsf{params}, h_1, \ldots, h_d, f(x))$$

does not have root equal to the c sent by the prover in the execution phase. Although c is sent only after the verifier samples ν, an adversary for ARG can internally: run the whole PoS initialization, sample ν, run $\mathcal{A}_2$ to obtain (c, π), and output the false root-checking instance together with π. Therefore

$$\Pr[\mathsf{FalseArg}] \le \varepsilon_{\mathsf{arg}}.$$

Let $\mathsf{TrueAcc}$ be the event that V_2 accepts and $\mathsf{FalseArg}$ does not occur. Then

$$\Pr[\mathsf{TrueAcc}] > \mu - \varepsilon_{\mathsf{arg}} > 2\theta_\rho.$$

For a fixed input x, let $q_x = \Pr[\mathsf{TrueAcc} \mid x]$. Define $X_{\mathsf{good}} = \{x : q_x > \theta_\rho\}$. By Lemma 9, applied with threshold $2\theta_\rho$, the set X_{good} has density greater than θ_ρ, and hence greater than $\varepsilon_{\mathsf{inc}}$. Fix any $x \in X_{\mathsf{good}}$ and write $y = f(x)$.

We now instantiate Lemma 11. The extraction initialization algorithm runs $\mathcal{A}_1$ on $(x, \mathbf{h})$ and outputs its stored state. The query algorithm, on a tuple ν, runs $\mathcal{A}_2$ on that state after sending ν, parses the response as $(c, \pi, \{(a_j, \mathbf{p}_j)\}_{j\in[k]})$, discards c and π, and outputs the claimed symbols and Merkle paths. Whenever $\mathsf{TrueAcc}$ occurs, the argument statement is true, so the root c sent by the prover is exactly the honest root of $y = f(x)$ under $\mathbf{h}$, and all k Merkle openings verify under this honest root. The bad-opening probability required by Lemma 11 is bounded by $k\varepsilon_{\mathsf{mt}}$, by the 2-PLOSC security of MT and a union bound over the k opened positions. Thus the lemma applies with success parameter θ_ρ and residual density

$$\gamma_\rho := \theta_\rho - k\varepsilon_{\mathsf{mt}}.$$

Consequently, there exists an advice z such that for every $x \in X_{\mathsf{good}}$, there is a string $\hat{y}$ such that $\Delta(y, \hat{y}) \le \beta n$ and

$$\mathsf{aK}^{t_K}_z(\hat{y} \mid x) \le (s - \Delta_\rho) + \Delta_\rho \le s.$$

Since X_{good} has density greater than $\varepsilon_{\mathsf{inc}}$ and $\beta < \beta + \eta$, this contradicts the assumed approximate incompressibility of f. Hence no such prover exists. □

Corollary 5 (Tight PoS from SNARGs)**.** *Assume the hypotheses of Theorem 7. Instantiate f using Corollary 4 so that it is $2/\sqrt{\log n}$-approximately incompressible against descriptions of length $n - O(n \log\log n/\sqrt{\log n})$, with negligible error. Assume also that the argument and Merkle soundness terms are negligible and that extraction runs within the incompressibility time bound. Then Construction 1 is sound against every time-τ prover storing at most*

$$n - C\frac{n \log\log n}{\sqrt{\log n}} = (1 - o(1))n$$

bits, for a sufficiently large constant C, with only negligible accepting probability.

6.2 Construction from Interactive Arguments

The second instantiation replaces the SNARG by a Kilian-style PCP argument. This reduces the cryptographic assumption on the argument system to collision-resistant hashing, but the honest prover must store the PCP string for the root-checking computation which has size quasipolynomial in the time to compute f.

Construction 2 (PoS from Kilian-style arguments)**.** *Let MT_1 be the 2-PLOSC scheme used to commit to $y = f(x)$, and let MT_2 be the position-binding vector-commitment used in Kilian's protocol for committing to a PCP string. Let $\mathsf{PCP} = (P, V)$ be a PCP for the relation R_{root} whose instances are*

$$u = (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c_1), \mathsf{cf}' = \mathsf{accept}, T_f),$$

and whose witnesses are accepting computations of M_f.

- ***Initialization phase.***

Verifier $V_1(\mathsf{params})$*:*

1. *Sample* $x \leftarrow \{0,1\}^{\ell_{\mathsf{in}}}$ *and hash functions* $\mathbf{h} = (h_1, \ldots, h_d)$ *for* MT_1.
2. *Send* $(x, \mathbf{h})$ *to* P_1.
3. *Output verifier state* $\Phi = (x, \mathbf{h})$.

Prover $P_1(\mathsf{params})$*:*

1. *Receive* $(x, \mathbf{h})$ *and compute* $y = f(x)$.
2. *Compute* $(c_1, \mathsf{st}_1) = \mathsf{MT}_1.\mathsf{Commit}(\mathsf{params}, h_1, \ldots, h_d, y)$.
3. *Form* $u = (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c_1), \mathsf{cf}' = \mathsf{accept}, T_f)$ *and compute the PCP string* $\Pi \leftarrow \mathsf{PCP}.P(u, w)$*, where* w *is the accepting computation of* M_f.
4. *Compute* $(c_2, \mathsf{st}_2) = \mathsf{MT}_2.\mathsf{Commit}(\mathsf{params}, \Pi)$.
5. *Store* $S = (c_1, c_2, \mathsf{st}_1, \mathsf{st}_2, \Pi)$.

- ***Execution phase.***

Verifier $V_2(\mathsf{params}, \Phi)$*:*

1. *Parse* $\Phi = (x, \mathbf{h})$*. Sample and send* $\nu = (i_1, \ldots, i_k) \leftarrow [n]^k$.
2. *Receive commitments* c_1, c_2 *and openings* $\{(y_{i_j}, \mathbf{p}_j)\}_{j \in [k]}$.
3. *For every* $j \in [k]$*, check* $\mathsf{MT}_1.\mathsf{Verify}(\mathsf{params}, c_1, i_j, y_{i_j}, \mathbf{p}_j) = 1$.
4. *Form* $u = (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c_1), \mathsf{cf}' = \mathsf{accept}, T_f)$*. Sample PCP randomness* ρ *and send it to* P_2.
5. *Receive* $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf})$.
6. *Check that* $\mathsf{MT}_2.\mathsf{Verify}(\mathsf{params}, c_2, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$ *and that* $\mathsf{PCP}.V^{[\mathcal{Q}, \mathsf{ans}]}(u; \rho) = 1$*, where the verifier rejects if the PCP verifier queries outside* $\mathcal{Q}$.
7. *Accept if and only if all checks accept and the response arrives within the execution time bound.*

Prover $P_2(\mathsf{params}, S)$*:*

1. *Parse* $S = (c_1, c_2, \mathsf{st}_1, \mathsf{st}_2, \Pi)$ *and receive* $\nu = (i_1, \ldots, i_k)$.
2. *For every* $j \in [k]$*, compute* $\mathbf{p}_j \leftarrow \mathsf{MT}_1.\mathsf{Open}(\mathsf{params}, \mathsf{st}_1, c_1, i_j)$.
3. *Send* c_1, c_2 *and* $\{(y_{i_j}, \mathbf{p}_j)\}_{j \in [k]}$.
4. *Receive PCP randomness* ρ*. Compute the PCP query set* $\mathcal{Q}$ *made by* $\mathsf{PCP}.V(u; \rho)$ *and compute* $\mathsf{pf} \leftarrow \mathsf{MT}_2.\mathsf{Open}(\mathsf{params}, \mathsf{st}_2, c_2, \mathcal{Q})$.
5. *Send* $(\mathcal{Q}, \mathsf{ans} = \Pi[\mathcal{Q}], \mathsf{pf})$.

Theorem 8 (Soundness from Kilian-style arguments)**.** *Assume the derandomization hypothesis required by Lemma 11.*

Let $k = \Theta(\log^2 n)$ and $\beta = 1/\sqrt{\log n}$ be the parameters fixed above.

Let $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^n$ be computable in time T_f and be $(\beta+\eta)$-approximately (t_{K}, s)-incompressible with error $\varepsilon_{\mathsf{inc}}$. Suppose MT_1 is $\varepsilon_{\mathsf{mt}}$-secure and the Kilian protocol obtained from PCP and MT_2 has soundness error $\varepsilon_{\mathsf{kil}}$ for relation R_{root}.

For every execution time bound τ and every parameter $\rho > 0$, let Δ_ρ denote the additive storage loss guaranteed by Lemma 11, at radius β and query count k, when applied with success parameter $\theta_\rho = \varepsilon_{\mathsf{inc}} + \rho/2$, Merkle error $\varepsilon_{\mathsf{mt}}$, second-phase time τ, and state-size bound $m \le s$. Suppose this application of Lemma 11 is in its stated parameter range and has decompressor running time at most t_{K}.

Then Construction 2, instantiated with k execution queries, is complete, has efficient verification and honest proving time polynomial in $T_f + n + |\Pi|$, and is

$$(s - \Delta_\rho,\ \tau,\ \varepsilon_{\mathsf{kil}} + 2\varepsilon_{\mathsf{inc}} + \rho)\text{-}\textit{sound}.$$

The honest prover's persistent state is $n + |\Pi| + \mathsf{polylog} n$ bits which is dominated by the proof length of $O(T\,\mathsf{poly}\log T)$.

Proof. Completeness and efficiency follow from correctness of the two commitments and the PCP. The only point about message order is that c_1 and c_2 are sent after the Merkle index tuple but before the PCP randomness ρ; hence the Kilian commitment to the PCP string is fixed before the PCP challenge, exactly as required in the standard soundness proof.

The soundness proof is parallel to the proof of Theorem 7. Let a cheating prover with state at most $s - \Delta_\rho$ and execution time at most τ make the verifier accept with probability

$$\mu > \varepsilon_{\mathsf{kil}} + 2\varepsilon_{\mathsf{inc}} + \rho.$$

Set $\theta_\rho = \varepsilon_{\mathsf{inc}} + \rho/2$. Let $\mathsf{FalseKil}$ be the event that the verifier accepts although the root-checking instance $u = (M_f, \mathsf{cf} = (\mathsf{params}, x, \mathbf{h}, c_1), \mathsf{cf}' = \mathsf{accept}, T_f)$ is false.

This event breaks the Kilian argument with the same probability up to its soundness error. The reduction samples the PoS initialization and the Merkle tuple ν, runs the cheating prover until it sends c_1, c_2 and the claimed Merkle openings, sets the Kilian instance to the corresponding u, sends c_2 as the PCP commitment, and then forwards the external PCP randomness to the prover. Whenever $\mathsf{FalseKil}$ occurs, the reduction convinces the Kilian verifier of a false instance. Therefore

$$\Pr[\mathsf{FalseKil}] \le \varepsilon_{\mathsf{kil}}.$$

Let $\mathsf{TrueAcc}$ be acceptance outside $\mathsf{FalseKil}$. Then $\Pr[\mathsf{TrueAcc}] > 2\theta_\rho$.

As before, define $q_x = \Pr[\mathsf{TrueAcc} \mid x]$ and $X_{\mathsf{good}} = \{x : q_x > \theta_\rho\}$. By Lemma 9, this set has density greater than θ_ρ, and hence greater than $\varepsilon_{\mathsf{inc}}$. For each fixed $x \in X_{\mathsf{good}}$, set $y = f(x)$ and invoke Lemma 11 with an initialization algorithm that runs the cheating initialization and a query algorithm that sends the tuple ν, receives c_1, c_2 and the Merkle openings, discards the PCP transcript, and outputs the claimed symbols and paths. On every $\mathsf{TrueAcc}$ execution, the root-checking instance is true, so c_1 is the honest Merkle root of $f(x)$ under $\mathbf{h}$, and the displayed openings verify under the honest root. The only remaining failure mode for the extraction lemma is a bad opening, bounded by $k\varepsilon_{\mathsf{mt}}$ by 2-PLOSC security of MT_1. Thus the extraction lemma is invoked with residual density

$$\gamma_\rho = \theta_\rho - k\varepsilon_{\mathsf{mt}}.$$

Lemma 11 therefore gives that there exists an advice string z, under which for every $x \in X_{\mathsf{good}}$, there exists a string $\hat{y}$ with $\Delta(f(x), \hat{y}) \leq \beta n$ and $\mathsf{aK}_z^{t_K}(\hat{y} \mid x) \leq s$. Since X_{good} has density greater than $\varepsilon_{\mathsf{inc}}$, this contradicts the $(\beta + \eta)$-approximate incompressibility of f. □

Remark 4. Since M_f runs in time T_f, quasilinear-size PCPs [BSS08, Din07] for R_{root} give proof length $|\Pi| = T_f^{1+o(1)}$ with $\mathsf{polylog}(T_f)$ queries and randomness and constant soundness error. Repeating the challenge $\lambda = \omega(\log n)$ times against the same commitment c_2 (fixed before ρ by the message ordering of Construction 2) yields Kilian soundness $\varepsilon_{\mathsf{kil}} = \mathsf{negl}(n)$, with proof length unchanged.

Corollary 6 (Kilian space bound with almost-linear PCPs)**.** *Instantiate the PCP in Construction 2 as in Remark 4, with negligible Kilian soundness. Under the same $\beta = 1/\sqrt{\log n}$ parameters as Corollary 5, every time-τ cheating prover storing at most*

$$n - C\,\frac{n \log\log n}{\sqrt{\log n}}$$

bits succeeds with only negligible probability. If N_{hon} is the honest prover's persistent storage, then

$$N_{\mathsf{hon}} = n + T_f^{1+o(1)} + \mathsf{polylog}(n).$$

Thus if $T_f = n^{1+o(1)}$ then an adversarial storage lower bound $N_{\mathsf{hon}}^{1-o(1)}$; more generally, $T_f = n^a$ gives $N_{\mathsf{hon}}^{1/a-o(1)}$.

References

[AAC+17] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 357–379. Springer, Cham, December 2017.

[AASY15] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions (extended abstract). In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, Portland, Oregon, USA, June 17-19, 2015*, LIPIcs, pages 582–600. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[ABB22] Mohammad Hassan Ameri, Alexander R. Block, and Jeremiah Blocki. Memory-hard puzzles in the standard model with applications to memory-hard functions and resource-bounded locally decodable codes. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 45–68. Springer, Cham, September 2022.

[ABFG14] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 538–557. Springer, Cham, September 2014.

[ABP17] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Cham, April / May 2017.

[ABP18] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130. Springer, Cham, April / May 2018.

[ACK+16] Joël Alwen, Binyi Chen, Chethan Kamath, Vladimir Kolmogorov, Krzysztof Pietrzak, and Stefano Tessaro. On the complexity of scrypt and proofs of space in the parallel random oracle model. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 358–387. Springer, Berlin, Heidelberg, May 2016.

[Adl78] Leonard Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 75–83. IEEE Computer Society, 1978.

[AFvMV06] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: Concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006.

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity (CCC 2016)*, pages 9–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

[AR23] Noga Amit and Guy N. Rothblum. Constant-round arguments from one-way functions. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1537–1544. ACM Press, June 2023.

[BGDM23] Marshall Ball, Eli Goldin, Dana Dachman-Soled, and Saachi Mutreja. Extracting randomness from samplable distributions, revisited. In *64th FOCS*, pages 1505–1514. IEEE Computer Society Press, November 2023.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016*, pages 345–356. ACM, January 2016.

[BPSS23] Nir Bitansky, Omer Paneth, Dana Shamir, and Tomer Solomon. Non-interactive universal arguments. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 132–158. Springer, Cham, August 2023.

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

[CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pages 68–79. IEEE Computer Society Press, February 2022.

[CT22] Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *62nd FOCS*, pages 125–136. IEEE Computer Society Press, February 2022.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Berlin, Heidelberg, August 2015.

[Din07] Irit Dinur. The pcp theorem by gap amplification. *Journal of the ACM (JACM)*, 54(3):12–es, 2007.

[DS26] Alon Dermer and Ronen Shaltiel. Multiplicative pseudorandom generators for nondeterministic circuits. *Electron. Colloquium Comput. Complex.*, TR26, 2026.

[Fis19] Ben Fisch. Tight proofs of space and replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Cham, May 2019.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C Oliveira. Probabilistic kolmogorov complexity with applications to average-case complexity. In *37th Computational Complexity Conference (CCC 2022)*, pages 16–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[GR07] Venkatesan Guruswami and Atri Rudra. Better binary list-decodable codes via multilevel concatenation. In Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, Lecture Notes in Computer Science, pages 554–568. Springer, 2007.

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *32nd ACM STOC*, pages 181–190. ACM Press, May 2000.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 520–549. Springer, Cham, May / June 2022.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005.

[KvMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Comput. Complex.*, 21(1):3–61, 2012.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Berlin, Heidelberg, August 1988.

[NW88] Noam Nisan and Avi Wigderson. Hardness vs. randomness (extended abstract). In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 2–11. IEEE Computer Society, 1988.

[RD16] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 262–285. Springer, Berlin, Heidelberg, October / November 2016.

[Rey24] Leonid Reyzin. Proofs of space with maximal hardness. In *65th FOCS*, pages 1159–1177. IEEE Computer Society Press, October 2024.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

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

[Sha25] Ronen Shaltiel. Multiplicative extractors for samplable distributions. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference, CCC 2025, Toronto, Canada, August 5-8, 2025*, LIPIcs, pages 22:1–22:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[SU01] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd FOCS*, pages 648–657. IEEE Computer Society Press, October 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 212–226. IEEE Computer Society, 2005.

[TV00] Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *41st FOCS*, pages 32–42. IEEE Computer Society Press, November 2000.