Electronic Colloquium on Computational Complexity, Report No. 116 (2024)

Fiat-Shamir in the Plain Model from Derandomization

Or: Do Efficient Algorithms Believe that $\mathcal{NP} = \mathcal{PSPACE}$?

Lijie Chen* Ron D. Rothblum† Roei Tell ‡

July 6, 2024

Abstract

A classical challenge in complexity theory and cryptography is to simulate interactive proof systems by *non-interactive* proof systems. In this work we leverage approaches from recent works in derandomization to address this challenge, focusing on non-interactive simulations that are sound against *uniform adversarial algorithms*.

Our results concern fundamental questions in **complexity theory**, such as the $\mathcal{NP}$ vs $\mathcal{PSPACE}$ question, and also in **cryptography**, such as the question of constructing non-interactive zero-knowledge arguments for $\mathcal{NP}$ from unstructured assumptions. Relying on strong complexity-theoretic hardness assumptions (that will be described below):

1. **Complexity theory.** We prove that $\mathcal{PSPACE}$ is contained in the "computationally sound" version of $\mathcal{NP}$. Specifically, for every $L \in \mathcal{PSPACE}$, membership in L can be verified by an $\mathcal{NP}$-type (deterministic, polynomial-time) verifier V with the following guarantee: The verifier accepts every $x \in L$ when given a proof π from an honest prover that runs in fixed exponential time T_P; and every uniform adversary running in probabilistic time $\text{poly}(T_P)$ cannot find $x \notin L$ and π such that $V(x, \pi) = 1$, except with negligible probability in T_P. As a corollary in the area of bounded arithmetic, under the same assumptions, we deduce that $\mathcal{NP} \neq \mathcal{PSPACE}$ is not provable in the theory APC_1. This is a strong theory, which captures many of the major results in complexity.
2. **Cryptography.** We construct new cryptographic protocols, including succinct non-interactive arguments (SNARGs) for $\mathcal{NC}$ in the plain model, as well as non-interactive zero-knowledge and witness indistinguishable (NIZK and NIWI) proof systems for $\mathcal{NP}$, all with computational soundness against uniform adversaries. The SNARG relies solely on the aforementioned complexity-theoretic assumption, whereas the NIZK and NIWI require also a sub-exponentially secure one-way function (which should be injective in the case of the NIWI). These are the first constructions of the above protocols that do not rely on highly structured cryptographic primitives.

Roughly speaking, following Chen and Tell (FOCS 2021, STOC 2023), the complexity-theoretic hardness assumptions throughout our paper assert the existence of functions $f\colon \{0,1\}^n \to \{0,1\}^k$ that are computable in polynomial time and hard for bounded-space machines (say, linear space) in a strong average-case sense: No efficient algorithm can find an input x on which the bounded-space machine computes f, except with negligible probability.

*Miller Institute for Basic Research in Science, University of California, Berkeley. Email: wjmzmbr@gmail.com
†Technion, Taub Faculty of Computer Science. Email: rothblum@cs.technion.ac.il
‡The University of Toronto, Department of Computer Science. Email: roei@cs.toronto.edu

ISSN 1433-8092

Contents

1 Introduction

In this work we leverage approaches from recent works in derandomization [CT21; CT23] to address the classical challenge of simulating interactive proof systems by procedures in which there is less interaction. This challenge has been studied for decades in complexity theory and in cryptography, and indeed the current work lies in the intersection of the two areas.

In **complexity theory**, the focus has been on proof systems with constantly many rounds of interaction. Unconditional simulations of such systems by proof systems with just one round of interaction (aka $\mathcal{AM}[2]$), at a cost of a polynomial time overhead, have been known since the 1980s (see [BM88; GS89], and also see the related [Les22]). Moreover, full derandomizations of constant-round proof systems (into $\mathcal{NP}$ proof systems) have been deduced from various hardness assumptions, in a rich and ongoing line of work (see, e.g., [KM02; MV05; SU05; SU07; CT23; MS23a; MS23b]). Indeed, recall that derandomization is a stronger result than round reduction, since eliminating the randomness removes the need for interaction (i.e., the prover can send the entire interaction, which is fully predictable, in advance).

In **cryptography**, the Fiat-Shamir heuristic [FS86] simulates certain classes of public-coin proof systems with arbitrarily many rounds by protocols that use just a single round of interaction, which in cryptography is called the CRS model.[1] In a nutshell, the idea is to apply a cryptographic hash function to the transcript at each round, and use the result as random coins for the verifier. A recent exciting line of work has shown how to instantiate this heuristic securely from a variety of cryptographic assumptions (e.g., under Learning With Errors (LWE)), compiling rich classes of proof systems into protocols that are secure against computationally bounded adversaries (aka argument systems; see below) and that are sometimes also *zero knowledge*; see, e.g. [KRR17; CCR+18; CCH+19; PS19; JKK+21; CJJ21]. Compared to the complexity-theoretic results, the classes of proof systems that can be simulated are richer (e.g., they contain systems with super-constantly many rounds). However, the assumptions are cryptographic, the simulation has both randomness and (a mild form of) interaction, and soundness is only computational.

A complexity-theoretic variant of Fiat-Shamir. Chen and Tell [CT23] recently showed that, under complexity-theoretic assumptions, constant-round proof systems can be derandomized into *deterministic argument systems* (see Definition 1.1 below). Their focus was on the classical setting for derandomization (i.e., of constant-round proof systems), and their goal was to minimize (in fact, eliminate) the computational overhead incurred by derandomization.

As noted in [CT23, Section 1.2.4], their derandomization techniques can be considered as a *complexity-theoretic variant of the Fiat-Shamir heuristic*. In a gist, instead of applying a cryptographic hash function to the transcript at each round to obtain coins for the verifier, they applied a targeted hitting-set generator (targeted HSG) to the transcript at each round to obtain a list of candidate choices for the verifier's random coins (and the non-interactive protocol was obtained by enumerating over all candidates, across all rounds). However, unlike works in cryptography studying the heuristic, their assumptions are *complexity-theoretic* (e.g., they are not known to imply one-way functions), and they derandomize proof systems into the *plain model* (rather than the CRS model), obtaining protocols that are sound against uniform probabilistic adversaries.

[1] In this model, a common random string (CRS) is chosen before the interaction, and provided to the prover and to the verifier. In complexity-theoretic terms, this is a one-round public-coin protocol in which the verifier speaks first.

Our contributions. The work of [CT23] naturally raises the following questions: Inspired by the reach of the Fiat-Shamir heuristic in cryptography, can we extend the complexity-theoretic approach of [CT23] so that it applies to richer classes of proof systems (in particular, to systems with super-constantly many rounds)? And can we use their techniques to simulate proof systems by non-interactive protocols that are zero knowledge (in the CRS model)?

The current work provides a surprisingly strong positive answer to both questions, and deduces significant consequences. Under complexity-theoretic assumptions that are similar in form to the ones introduced in [CT23], we simulate rich classes of interactive proofs (including ones with polynomially many rounds) by argument systems that are **deterministic** (in particular, non-interactive), or that are in the CRS model and are **zero knowledge** (when assuming also mild cryptographic assumptions). Our results have implications to the $\mathcal{NP}$ vs $\mathcal{PSPACE}$ question, to the challenge of constructing non-interactive zero-knowledge arguments for $\mathcal{NP}$, and to the question of which fragment of bounded arithmetic is needed to prove $\mathcal{NP} \neq \mathcal{PSPACE}$.

Furthermore, we show that the connection between derandomization-based techniques and the Fiat-Shamir heuristic is **inherent**, rather than an artifact of specific proofs. Specifically, we prove an *equivalence* between a natural relaxed version of correlation intractable hash functions (CIHFs), which are the objects commonly used to instantiate the Fiat-Shamir heuristic; and targeted hitting-set generators (targeted HSGs), which are the objects built in [CT21; CT23] (following Goldreich [Gol11]) for non-black-box derandomization of proof systems. The aforementioned natural relaxation of CIHFs allows the function to output a list of candidates, and we prove that this relaxation still suffices for applying the Fiat-Shamir heuristic in several prominent settings.

1.1 Technical setup: The uniform setting, assumptions, and key observation

Recall that argument systems, first defined by Brassard *et al.* [BCC88], are proof systems in which soundness is not information-theoretic, but rather computational: No efficient adversary can mislead the verifier. We will be interested in argument systems in which the verifier is a *deterministic* $\mathcal{NP}$-type procedure, and the adversary is a *uniform* (probabilistic) Turing machine.[2]

Definition 1.1 (computationally sound $\mathcal{NTIME}$)**.** *We say that a function $f\colon \{0,1\}^* \to \{0,1\}^*$ is in* computationally sound non-deterministic time T_V with a time-T_P prover*, denoted*

$$f \in \text{cs-}\mathcal{NTIME}[T_V, T_P] \,,$$

if there exist a uniform verifier V and a uniform honest prover P such that:

1. **Completeness.** *For every $x \in \{0,1\}^*$ it holds that $V(x, P(x)) = f(x)$.*
2. **Soundness.** *For every sufficiently large $n \in \mathbb{N}$ and every uniform probabilistic adversary $\tilde{P}$ running in time polynomial in T_P, the probability that $\tilde{P}(1^n)$ prints (x, π) such that $|x| = n$ and $V(x, \pi) \notin \{f(x), \perp\}$ is negligible in $T_P(n)$.*
3. **Efficiency.** *On input $x \in \{0,1\}^*$ the honest prover P runs in deterministic time $T_P(|x|)$. On input (x, π), the verifier runs in deterministic time $T_V(|x|)$.*

[2] This notion coincides with what was defined in [CT23] as "deterministic effective argument systems". The more natural name "computationally sound $\mathcal{NP}$" was later suggested by Oded Goldreich. Also note that our notion is related to Micali's [Mic00] "computationally sound proofs", but we emphasize that we focus on the *plain model*, whereas Micali's notion is in the random oracle model.

The definition extends to partial functions in the natural way. For Boolean functions, represented as promise-problems $\Pi = (\mathsf{Y}, \mathsf{N})$, we require completeness to hold only with respect to $x \in \mathsf{Y}$ and soundness to hold only with respect to finding (x, π) such that $x \in \mathsf{N}$ and $V(x, \pi) = 1$.

Note that Definition 1.1 is fully uniform. In particular, the adversary $\widetilde{P}$ is a uniform algorithm, and hence soundness is guaranteed only on inputs that can be efficiently sampled.[3] Definition 1.1 bounds the adversary's runtime to be polynomial in the honest prover's runtime, but a more general definition (allowing the adversary more resources) would also be reasonable.

Hardness assumptions. We will show how to simulate interactive proof systems in cs-$\mathcal{NTIME}$, based on assumptions of the following form: There is a function f with multiple output bits that is computable in polynomial time but hard to compute in smaller space (say, space n^{ϵ}), where the crucial point is that the hardness holds when the input x comes from any efficiently samplable distribution. This is a strong form of average-case hardness, which can be equivalently thought of as asserting that: it is hard to find an input x on which f is easy. For example:

Assumption 1.2 (hardness over all samplable distributions, an illustrative assumption)**.** *There exists $f\colon \{0,1\}^* \to \{0,1\}^*$ mapping n bits to $n^{o(1)}$ bits such that f is computable in polynomial time and satisfies the following. For every probabilistic algorithm* Samp *running in (arbitrarily large) polynomial time, and every (deterministic) space-$n^{.01}$ algorithm A, and every sufficiently large $n \in \mathbb{N}$ it holds that*

$$\Pr_{x \leftarrow \mathsf{Samp}(1^n)}[A(x) = f(x)] \leq \mathsf{negl}(n) .$$

We will refer to hardness as in Assumption 1.2 as hardness over all efficiently samplable distributions. Assumptions of hardness over all efficiently samplable distributions have been used in recent years to deduce "free lunch" derandomization (i.e., derandomization with essentially no time overhead) of probabilistic algorithms [CT21] and of constant-round proof systems [CT23].

Indeed, a random function attains the hardness conjectured in Assumption 1.2 (with high probability), and this will be true for all of our assumptions in the paper. Hence, it is plausible to assume that an explicit function also has the required hardness. (For concrete instantiations in practical applications it seems possible to consider, e.g., the SHA-256 function.) For further discussions of the assumption see the discussion after Theorem 1.3, and also Sections 1.5 and 2.2.

The motivating technical observation. From a technical perspective, the motivating observation underlying this work is:

> A function as in Assumption 1.2 *already suffices* to instantiate the Fiat-Shamir heuristic (equivalently, for non-black-box derandomization of proof systems), when working against uniform computationally-bounded adversaries.

To elaborate, in works studying the Fiat-Shamir heuristic, a cryptographically hard function (say, LWE as in [CJJ21]) is used to construct a CIHF, and the latter is used to instantiate the Fiat-Shamir heuristic. In works studying non-black-box derandomization, a function hard over all

[3] The restriction that $\widetilde{P}$ will be uniform is crucial. For example, if $\mathcal{NP} \neq \mathcal{PSPACE}$, then for every $\mathcal{NP}$-verifier V and $L \in \mathcal{PSPACE}$ there is either (1) Some $x \in L$ that the verifier rejects regardless of π, or (2) Some $x \notin L$ and π that the verifier accepts. A non-uniform adversary may simply have these hard-wired. Also recall that in any argument system (i.e., more generally than just for cs-$\mathcal{NTIME}$), to get a non-trivial definition we must allow the adversary $\tilde{P}$ to use strictly more resources than the honest prover P (see Section 3.4).

efficiently samplable distributions (say, a non-batch-computable function as in [CT23]) is used to construct a targeted HSG, and the latter is used for derandomization.

In contrast, in this work, the function f from Assumption 1.2 *will be used as-is* for non-black-box-derandomization (equivalently, for Fiat-Shamir), without a need for an additional construction of a CIHF or a targeted HSG from f.

The function f can be thought of as a CIHF for searchable relations (in the sense of Canetti *et al.* [CCH+19]) computable in small space, with soundness against *uniform adversaries*; we elaborate on this in Section 2. Indeed, one may view the key technical observation as follows: When working in the *uniform setting*, a CIHF for searchable relations that suffices for Fiat-Shamir is, essentially, a complexity-theoretic hardness assumption of the type introduced in [CT21; CT23].[4] Moreover, as mentioned above, this turns out not to be a coincidence, and we show that the connection between Fiat-Shamir and derandomization-based techniques is inherent (see Section 1.4).

1.2 Do efficient algorithms believe that $\mathcal{NP} = \mathcal{PSPACE}$?

Our first result is that under an assumption of the form similar to that of Assumption 1.2, we have (loosely speaking) that $\mathcal{PSPACE} \subseteq \text{cs-}\mathcal{NP}$; that is, every problem in $\mathcal{PSPACE}$ can be decided by an $\mathcal{NP}$-verifier that cannot be misled by any efficient adversary.

In fact, we show a full equivalence between the latter conclusion and a corresponding hardness assumption. Specifically, consider a relaxation of the upper bound in Assumption 1.2, where instead of assuming that the hard function f is computable in polynomial time (i.e., in $\mathcal{FP}$), we only assume that f is computable in $\text{cs-}\mathcal{NTIME}[\text{poly}(n), 2^n]$. We show that this assumption is *sufficient and necessary* for $\mathcal{PSPACE} \subseteq \text{cs-}\mathcal{NP}$; that is:

Theorem 1.3 ($\mathcal{PSPACE} \subseteq \text{cs-}\mathcal{NP}$; see Theorem 4.3)**.** *The following two statements are equivalent:*

1. *For some sufficiently small $\epsilon > 0$, and for some $\delta > 0$, there is $f\colon \{0,1\}^n \to \{0,1\}^{n^\epsilon}$ in $\text{cs-}\mathcal{NTIME}[\text{poly}(n), 2^{n^\delta}]$ that is hard for space $n^{c\cdot\epsilon}$ over all $2^{O(n^\delta)}$-time samplable distributions, where $c > 1$ is a universal constant.*
2. *For every $L \in \mathcal{PSPACE}$ there are $C > c > 1$ such that $L \in \text{cs-}\mathcal{NTIME}[n^C, 2^{n^c}]$.*

The hardness in Item (1) of Theorem 1.3 is attained by a random function $\{0,1\}^n \mapsto \{0,1\}^{n^\epsilon}$, with high probability, when $\delta < \epsilon$ (i.e., such a function is indeed hard for bounded-space machines over all distributions samplable in uniform time $2^{O(n^\delta)}$). In particular, it seems plausible that a function with such parameters exists even in $\mathcal{FP}$ or $\mathcal{FNP}$, rather than only in $\text{cs-}\mathcal{NP}$.

The second item in Theorem 1.3 suggests an additional perspective on the widely believed conjecture that $\mathcal{NP} \neq \mathcal{PSPACE}$: Even if the conjecture is true, the effective restriction that it places on $\mathcal{NP}$ verifiers is limited in its extent. This is because any $L \in \mathcal{PSPACE}$ can be decided by an $\mathcal{NP}$-verifier V_L that, from the perspective of algorithms running in fixed exponential time 2^{n^c}, works correctly.[5] To illustrate this, consider the perspective of V_L: It gets x and a proof π,

[4] The potential of uniform assumptions to yield a Fiat-Shamir heuristic secure only against uniform adversaries, on inputs that can be efficiently generated, was pointed out by Halevi, Myers, and Rackoff [HMR08, Section 5.3].

[5] The running time 2^{n^c} of the honest prover is not coincidental. Under reasonable assumptions, it cannot be significantly improved (see Section 3.4). On the other hand, allowing the prover to run in sufficiently large exponential time (i.e., larger than $2^{O(T_V)}$ where T_V is the verifier's running time) means that the adversary can perform a brute-force search over all witnesses; a system resilient against such adversaries is just a standard $\mathcal{NP}$-system.

and decides L correctly as far as any algorithm running in time comparable to that of V_L can tell. Also consider the perspective of the honest prover, which runs in time $N = 2^{n^c}$: It can convince an extremely quick deterministic V_L (i.e., running in time $\text{poly}(n) = \text{polylog}(N)$) that $x \in \{0,1\}^n$ belongs in L, without worrying that $\text{poly}(N)$-time adversaries might mislead V_L.

We emphasize that this statement should not appear unbelievable. Indeed, Kilian's [Kil92] classical result, when scaled up, implies that under cryptographic assumptions every problem in $\mathcal{NEXP}$ can be decided by a four-message interactive argument system in which the verifier runs in polynomial time (and has computational soundness).[6] The recent remarkable result of Choudhuri *et al.* [CJJ21] implies that, under standard cryptographic assumptions, every problem in $\mathcal{EXP}$ can be decided by a two-message polynomial-time argument system. A useful intuition is that if we could completely derandomize these argument systems, then we would obtain conclusions that are even stronger than the one in Theorem 1.3 (e.g., deduce $\mathcal{EXP} \subseteq \texttt{cs-}\mathcal{NP}$). Alas, the challenge of derandomizing these argument systems, which in the case of [Kil92] is decades old (see also [BBH+19] and references therein), remains open.[7]

We take a different route: In a gist, we "derandomize $\mathcal{IP} = \mathcal{PSPACE}$", with computational soundness. That is, we decide any $L \in \mathcal{PSPACE}$ by an interactive proof system, and the crux of our proof is simulating this system in $\texttt{cs-}\mathcal{NTIME}[\text{poly}(n), 2^{n^c}]$. (We use the system of [GKR15], which suffices for the $\mathcal{IP} = \mathcal{PSPACE}$ theorem [LFK+92; Sha92].) See Section 2.2 for details.

Remark 1.4. *The* $\texttt{cs-}\mathcal{NTIME}$ *system in Theorem 1.3 satisfies a soundness condition that is even stronger than stated, and holds for* all *inputs: For every adversary* $\widetilde{P}$ *running in time* $2^{O(n^c)}$*, and every (sufficiently long) input* $x \notin L$*, the probability that* $\widetilde{P}$ *finds* π *such that* $V(x,\pi) = 1$ *is negligible (in* 2^{n^c}*). This is because the system is sound against adversaries running in time larger than* $2^{O(n)}$*, which means that it is sound even against adversaries doing an exhaustive search over the inputs.*

1.2.1 Bounded arithmetic: How provable is $\mathcal{NP} \neq \mathcal{PSPACE}$?

What does it mean that efficient algorithms "believe that $\mathcal{NP} = \mathcal{PSPACE}$"? Under the assumptions of Theorem 1.3, there is an $\mathcal{NP}$-verifier that looks correct to all algorithms running in a certain fixed exponential time. However, there could still be a short proof of $\mathcal{NP} \neq \mathcal{PSPACE}$ that theoretical computer scientists (or their computers) could quickly read and verify.

Nevertheless, as pointed out to us by Ján Pich, a corollary of Theorem 1.3 is that under the same hardness assumption, any such proof will have to be "inherently complicated". That is, it will have to use a broad fragment of Peano arithmetic – seemingly broader than what has been used for most major results in complexity theory so far:

Theorem 1.5 ($\mathsf{APC}_1 \nvdash \mathcal{NP} \neq \mathcal{PSPACE}$; see Theorem 4.8)**.** *Under the hardness assumption of Item (1) in Theorem 1.3,* $\mathcal{NP} \neq \mathcal{PSPACE}$ *is not provable in* APC_1*.*

We will not attempt to define the theory APC_1 here; see, e.g., [Kra19, Section 12.6] for an excellent reference. However, we comment that this is a strong theory: Many major results in complexity theory are provable using only APC_1, including the PCP theorem [ALM+98] (see [Pic15b]),

[6] We compare our result to Kilian's argument system, rather than to Micali's [Mic00] computationally sound proofs, since – as mentioned in Footnote 2 – our $\texttt{cs-}\mathcal{NTIME}$ verifiers do not rely on a random oracle (indeed, they are just $\mathcal{NP}$-type machines, i.e. work in the plain model).

[7] We note that one could simply conjecture that, say, the [CJJ21] argument system is secure (against uniform adversaries) if the common random string is instantiated using a fixed string (e.g., the digits of π). Here we refer to constructions with meaningful security reductions.

hardness amplification for Boolean functions [STV01] (see [Jeř05, Section 4.3.5]), the connection between circuit lower bounds and pseudorandom generators [NW94; IW97] (see [Jeř07]), and classical lower bounds for constant-depth circuits and for monotone circuits [Raz85; Hås87; Raz87; Smo87] (see [Raz95; MP20]). A notable exception was provided in [CLO24], who showed that known lower bounds for very restricted computational models (i.e., one-way communication and single-tape Turing machines) are not provable in APC_1, under cryptographic assumptions. Lower bounds for *general models* that may be true and are known to be beyond the reach of APC_1 include strong lower bounds for non-uniform Σ_3-circuits (see [LO23], and also the related [Pic15a; PS21]). Indeed, Theorem 1.5 yields the first widely believed conjecture about general computational models that (under plausible assumptions) is not provable in APC_1.

1.2.2 Refuters: How constructive is $\mathcal{NP} \neq \mathcal{PSPACE}$?

For a language L and a class $\mathcal{C}_{\mathsf{weak}}$ of computational procedures, we say that a lower bound $L \notin \mathcal{C}_{\mathsf{weak}}$ is $\mathsf{constructive}$ if given $M_{\mathsf{weak}} \in \mathcal{C}_{\mathsf{weak}}$, we can efficiently find inputs on which M_{weak} errs in deciding L. An algorithm that finds such inputs is called a $\mathsf{refuter}$ for L against M_{weak}.

Refuters have been studied since (at least) the work of Kabanets [Kab01] (see also [LY94; Gut06; BTW10; Ver13; OS18; GSTS07; Ats06; DFG13; CTW23]). One of the main discoveries is that classical conjectured lower bounds for $\mathcal{P}$ and $\mathcal{BPP}$, such as $\mathcal{P} \neq \mathcal{NP}$ or $\mathcal{BPP} \neq \mathcal{NEXP}$, are *inherently constructive:* If the lower bound is true, then any M_{weak} (in $\mathcal{P}$ or $\mathcal{BPP}$, respectively) has a polynomial-time refuter (see [CJS+21, Theorem 1.2], following [GSTS07; BTW10]).

The meaning of Theorem 1.3 is that, under our assumptions, $\mathcal{NP} \neq \mathcal{PSPACE}$ (if true) is *inherently non-constructive*: For any $L \in \mathcal{PSPACE}$, there is an $\mathcal{NP}$-verifier that cannot be efficiently refuted, even in (fixed) exponential time. This naturally raises the question: What is the minimal complexity for refuting $\mathcal{NP} \neq \mathcal{PSPACE}$? Following [CJS+21], we prove:

Theorem 1.6 (constructivity of $\mathcal{NP} \neq \mathcal{PSPACE}$; see Theorem 4.5)**.** *Assume that $\mathcal{NP} \neq \mathcal{PSPACE}$. Then, for every $\mathcal{NP}$-machine M there exists a $\mathcal{P}^{\mathcal{NP}}$ machine Ref_M such that for infinitely many $n \in \mathbb{N}$ it holds that $\mathsf{Ref}_M(1^n)$ prints an n-bit input x such that M does not decide TQBF correctly on x.*

The complexity of Ref can be reduced further to $\mathcal{FNP}$, at the cost of providing Ref with $O(\log n)$ bits of non-uniform advice (see Theorem 4.5). Thus, under the assumptions of Theorem 1.3: If $\mathsf{TQBF} \notin \mathcal{NP}$, then there is an $\mathcal{NP}$-machine M that we cannot refute in (some) fixed exponential time, but we can do it in $\mathcal{P}^{\mathcal{NP}}$ or in $\mathcal{FNP}/O(\log n)$.

1.2.3 Fast computationally-sound $\mathcal{NP}$-verifiers for $\mathcal{NC}$

The proof approach underlying Theorem 1.3 (i.e., derandomizing the proof system of [GKR15] with computational soundness) can be applied to obtain more conclusions. As one application, we show that under an assumption of a similar flavor, uniform $\mathcal{NC}$ can be decided by $\mathtt{cs}$-$\mathcal{NTIME}$ verifiers running in near-linear time:

Theorem 1.7 (uniform-$\mathcal{NC} \subseteq \mathtt{cs}$-$\mathcal{NTIME}[n^{1+o(1)}]$; see Corollary 4.4)**.** *Let L be a problem in logspace-uniform $\mathcal{NC}$. Suppose that there is $f\colon \{0,1\}^n \to \{0,1\}^{\mathrm{polylog}(n)}$ in $\mathtt{cs}$-$\mathcal{NTIME}[n^{1+o(1)}, \mathrm{poly}(n)]$ such that f is hard for polylogarithmic space over all polynomial-time samplable distributions. Then, for every $\epsilon > 0$ it holds that $L \in \mathtt{cs}$-$\mathcal{NTIME}[n^{1+\epsilon}, \mathrm{poly}(n)]$.*

Compared to Theorem 1.3, in Theorem 1.7, we assume that the hard function is verifiable in near-linear time (rather than polynomial time), but the hardness is only for algorithms using space $\text{polylog}(n)$ (rather than linear space). The running time of the verifier is optimal, up to the constant $\epsilon > 0$, since we cannot decide all $\mathcal{NC}$ problems without reading the entire input.

Similarly to Theorem 1.3, the conclusion of Theorem 1.7 was not previously known to follow from natural hardness assumptions. Interactive proof systems (i.e., with interaction and statistical soundness) for logspace-uniform $\mathcal{NC}$ in which the verifier runs in time $\tilde{O}(n)$ were constructed by Goldwasser, Kalai, and Rothblum [GKR15], and SNARGs in the CRS model (for all of $\mathcal{P}$) were constructed from cryptographic assumptions in [KPY19; CCH+19; JKK+21; CJJ21; CGJ+23].[8] However, the only known non-interactive argument system in the *plain model* for a subclass of $\mathcal{BPP}$ was constructed for $\mathsf{DTISP}[\text{poly}(n), n^{\delta}]$, which is incomparable to uniform $\mathcal{NC}$ (see [CT23]).

1.3 Zero-knowledge from hardness over efficiently samplable distributions

Finding the minimal sufficient assumptions for cryptographic primitives and protocols is one of the holy grails of cryptography. Even for central notions, such as non-interactive zero knowledge (NIZK), we still have a relatively poor understanding of what form of assumptions is required: most notably, whether the existence of one-way functions suffices.

As our second family of applications, we show that *uniform versions* of several central notions can be based on very mild cryptographic assumptions (e.g., one way functions) combined with a complexity-theoretic hardness assumption as in Assumption 1.2 (i.e., hardness over all efficiently samplable distributions). Specifically, under such assumptions, for every $\mathcal{NP}$ language, we build:

- **Zaps and NIWI:** A two-message, and under a slightly stronger cryptographic assumption, even single-message, witness-indistinguishable (WI) argument system. The two-message variant is often called a zap [DN07] and the single-message variant is called a NIWI.
- **NIZK:** As a corollary, a non-interactive zero-knowledge (NIZK) argument system in the common random string (CRS) model.

Crucially, the protocols that we construct are secure against *uniform adversaries* (analogously to Definition 1.1). We define the relevant notions carefully and discuss them in Section 3.1, and throughout the current section, whenever we refer to NIZK, NIWI, or zap, we mean this model.

Results statements. Let us recall the definitions of the objects we construct, at a high level (formal definitions appear in Section 3.1). A NIWI is an $\mathcal{NP}$-style protocol that is *witness indistinguishable* (WI): For any input x and two $\mathcal{NP}$-witnesses w_1, w_2, the proof generated using w_1 is computationally indistinguishable from one generated using w_2. We emphasize that in contrast to standard interactive zero-knowledge proofs, a NIWI is fully non-interactive and does not require any form of setup (i.e., it is in the *plain* model rather than CRS model).[9] A zap is a 2-message public-coin WI argument system, where we emphasize that the WI property should

[8] Amit and Rothblum [AR23] showed that even assuming just one-way functions suffices to obtain argument systems for $\mathcal{NC}$ with a near-linear time verifier, although these systems use constantly many rounds of interaction.

[9] Indeed, as shown by Goldreich and Oren [GO94], interaction is inherently required for zero-knowledge. In contrast, for the weaker notion of WI, our work (as well as prior works which are discussed in Section 5.2) construct a NIWI consisting of a single message sent from the (probabilistic) prover to the (deterministic) verifier.

hold even wrt a malicious verifier. Lastly, NIZKs are non-interactive protocols with the full guarantee of zero-knowledge (i.e., the verifier's view can be simulated), but require a trusted setup – namely, a common random string (CRS) that is shared between the prover and verifier.

Prior constructions of such protocols all relied on *highly structured cryptographic assumptions*. Among the ones that have been used are (doubly-enhanced) trapdoor permutations, learning with errors (LWE), indistinguishability obfuscation (IO), and bilinear maps; see Section 5.2 for a detailed discussion and references. Our main result in this section is the following:

Theorem 1.8 (zaps and NIWI arguments for $\mathcal{NP}$)**.** *Assume that there are subexponentially secure one way functions. For $\epsilon > 0$ and a sufficiently small $\delta > 0$, assume that there is $f\colon \{0,1\}^n \to \{0,1\}^{n^\delta}$ in $\mathcal{FP}$ that is hard for linear-time algorithms with oracle access to $\mathcal{NTIME}[n^\epsilon]$ over all polynomial-time samplable distributions. Then, every $\mathcal{NP}$ relation R has a zap argument system.*

Assuming in addition a sub-exponentially secure non-interactive commitment scheme (i.e., statistically binding and computationally hiding), every $\mathcal{NP}$ relation R has a NIWI *argument system.*

Indeed, the main novelty of Theorem 1.8 is that it relies only on weak cryptographic assumptions (i.e., on one-way functions), and replaces the need for structured cryptographic hardness with the existence of an *unstructured* function f that is hard over all efficiently samplable distributions. The non-interactive commitment scheme in Theorem 1.8 (required for the NIWI result) can be instantiated using (sub-exponentially secure) *injective* one-way functions [Blu82], or assuming an additional NW-style derandomization assumption [OV07].

The closest related work is that of Barak *et al.* [OV07], who constructed NIWI assuming the existence of statistically sound zaps (in the uniform CRS model) and a complexity-theoretic assumption. Such zaps, however, are only known to exist under highly structured assumptions (e.g., doubly-enhanced trapdoor permutations). We discuss more prior works in Section 5.2.

Remark 1.9. *The linear-time algorithm with an $\mathcal{NTIME}$ oracle used in Theorem 1.8 is weaker than the $\mathcal{DSPACE}$ algorithm used in Theorem 1.3. Moreover, the oracle is only used in order to break a cryptographic commitment scheme, and can be replaced with a one-way function inverter.*

In fact, if we rely on a commitment scheme that can be broken by a quantum machine (e.g., one based on factoring), then we can omit the $\mathcal{NTIME}$ oracle altogether and replace it by hardness for a quantum machine trying to compute the function (on any classical polynomial-time samplable distribution).

Remark 1.10. *The complexity-theoretic assumption underlying Theorem 1.8 is not* falsifiable *(in the sense of Naor [Nao03]) because of the $\mathcal{NTIME}$ oracle, but the quantum variant mentioned in Remark 1.9 is falsifiable by a quantum machine.*

Obtaining a NIZK**.** Using the celebrated "FLS Trick" [FLS99] we can convert the protocols of Theorem 1.8 into a full-fledged non-interactive zero-knowledge (NIZK) argument system, in the CRS model (see Definition 3.3 for a careful definition of NIZK in the uniform setting).

Corollary 1.11 (NIZK)**.** *Assuming subexponentially secure one way functions and a function f as in Theorem 1.8, every $\mathcal{NP}$ relation R has a* NIZK *argument system in the* CRS *model.*

As mentioned above, constructing a NIZK from a "minicrypt" type of assumption is a major open problem. Corollary 1.11 makes progress towards this important goal by relying, in addition to the one-way function, on a (strong) complexity-theoretic hardness assumption.

1.4 The connection between Fiat-Shamir and derandomization is inherent

Recall that our assumptions of hardness over all efficiently samplable distributions arise from recent research into non-black-box derandomization (see [CT21; CT23]).

We show that the connection between the complexity-theoretic study of non-black-box derandomization and the cryptographic study of the Fiat-Shamir heuristic is *inherent*. To see this, let us recall the assumptions and the constructed algorithmic objects in both lines of work:

1. In the cryptographic works (e.g., [CCR16; KRR17; HL18; CCR+18; BKM20; JJ21; JKK+21; HLR21; CJJ21; HJK+22; CGJ+23]), cryptographic assumptions are used to construct correlation-intractable hash functions (CIHFs). These objects, first defined by Canneti, Goldreich, and Halevi [CGH04], are functions h such that for any sparse relation R from a certain class $\mathcal{R}$ of relations, it is infeasible to find an input x for which $(x, h(x)) \in R$.

2. In works studying non-black-box derandomization [CT21; CT23], hardness over all efficiently samplable distributions is used to construct targeted hitting-set generators (targeted HSGs) that are pseudorandom over efficiently samplable distributions. These are algorithms that get an input x coming from an efficiently samplable distribution, and produce a list of strings that "hits" every dense set S from a collection $\mathcal{S}_x$ of sets (see Definition 6.2).

In both lines of work, the object that is constructed (i.e., targeted HSG or CIHF) is used in a very similar way: it is applied to the transcript in each round to obtain the verifier's next message (or several candidate messages, in the case of targeted HSGs).

This is indeed not a coincidence: When modelling adversaries as uniform algorithms, targeted HSGs are *completely identical* to a natural relaxation of CIHFs, in which h is allowed to output a list of strings.[10] To illustrate this with a concrete setting, consider targeted HSGs that are pseudorandom for space S (i.e., on input x, the generator hits all dense sets from the collection $\mathcal{S}_x$ of sets decidable by uniform space-S algorithms with access to x); analogously, for CIHFs, consider the class $\mathcal{R}$ of all relations recognizable in uniform space S. Then:

Theorem 1.12 (informal; see Theorem 6.7)**.** *A function G is a targeted HSG that is pseudorandom for uniform space-S algorithms over a distribution $\mathbf{x}$ of inputs* if and only if *G is list-correlation-intractable over $\mathbf{x}$ for sparse relations recognizable by uniform space-S algorithms.*

One may wonder if the relaxation of CIHFs to list-CIHFs (which is needed for the equivalence) is significant in this context. Crucially, we prove that list-CIHFs *suffice for the target application* of Fiat-Shamir in important settings. In particular, we show that *all of the results* in the current paper can be deduced from list-CIHFs avoiding suitable relations, rather than only from assumptions a-la Assumption 1.2. (See Corollary 7.6 for a technical statement that list-CIHFs suffices for $\mathcal{PSPACE} \subseteq \text{cs-}\mathcal{NP}$ and Theorem 7.8 for a technical statement that list-CIHFs suffice to get NIZK for $\mathcal{NP}$.) While allowing a list is a natural relaxation, the fact that it suffices for the results is surprising; we elaborate on this in Section 2.1, and also explain which list size the results support.

The key takeaway is that when considering uniform adversaries, the algorithmic objects that have been considered in both lines of work *are, essentially, the same object*.

[10] In more detail, we can compare targeted HSGs to "keyless" list-CIHFs, and we can also compare a "keyed" version of targeted HSGs to standard ("keyed") list-CIHFs, and both comparisons yield an equivalence. To alleviate concerns, we note that "keyless" CIHFs may exist for the classes of relations we are interested in, despite the impossibility result of [CGH04] (because we are interested in CIHFs computable by algorithms that are allowed more time than the procedures that recognize the sparse set the CIHF needs to avoid; see Section 6).

Constructions of list-CIHFs from hardness on all efficiently samplable distributions. List-CIHFs that suffice for our results can be constructed from hardness on all efficiently samplable distributions, using a targeted PRG from [CT21; CT23]. Loosely speaking, suitable list-CIHFs follow from the existence of a function in $\mathcal{FP}$ that is *non-batch-computable* (in the sense of [CT21; CT23]) by algorithms with a non-trivial oracle (e.g., a linear-space oracle) over all polynomial-time samplable distributions. For details see Section 7.3 and Corollaries 7.15 and 7.16.

Indeed, this allows to deduce the consequences of Theorems 1.3 and 1.8 from different assumptions, i.e. from non-batch-computability over all efficiently samplable distributions. The latter assumptions seem somewhat less clean than the ones stated in Theorems 1.3 and 1.8.

Why was the equivalence not discovered before? The Fiat-Shamir heuristic has been widely influential in practice and extensively studied theoretically since its introduction in the 1980's. Similarly, derandomization is a central area in theoretical computer science, studied extensively since (at least) the late 1970's. The equivalence in Theorem 1.12 is thus important, and despite being almost immediate given the proper definitions, it was not known before.

The culprit is that in this work we consider the **uniform setting**. In contrast, both lines of research traditionally focused on non-uniform adversaries, in which case the two objects seem very different from each other. For example, classical PRGs for non-uniform circuits have seed length that is too large for typical Fiat-Shamir applications: The PRG can only derandomize proof systems with *constantly many rounds* (without super-polynomial overheads), and it is not compatible with constructions of NIZK for $\mathcal{NP}$ (because the list size of the corresponding list-CIHF is too large; see Section 2.1). And if we want a CIHF secure against non-uniform adversaries, we need a keyed function, and thus can only work in the CRS model (indeed, the corresponding derandomization object would be quite unnatural: a keyed targeted HSG with seed length zero).

1.5 More on the hardness assumptions

The non-standard part in Assumption 1.2 (and in all of our assumptions of this flavor) is the strong average-case hardness: the function f is hard not only over (say) the uniform distribution, but over any efficiently samplable distribution.

While this is a strong notion, we believe that it is quite plausible, and it is of independent interest. As mentioned after Assumption 1.2 and Theorem 1.3, such hardness is attained by a random function. Moreover, such average-case hardness is far from unprecedented in complexity theory, let alone in cryptography. For example, a function hard for probabilistic algorithms over all efficiently samplable distributions was recently used in [CT21; CT23] for "free lunch" derandomization of probabilistic algorithms and of constant-round proof systems, and hardness for probabilistic algorithms over all efficiently samplable distributions is in fact necessary for their derandomization conclusion (as is the assumption in Theorem 1.3 for the conclusion $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$). In fact, there are even, unconditionally, explicit functions that are hard on *all inputs* (except finitely many) for certain classes; see examples in [CT21].[11]

We pose the study of this type of average-case hardness assumption as a **major open problem**. Indeed, a natural suspicion is that such strong average-case hardness might imply other strong

[11] We comment, though, that known functions that are hard on all but finitely many inputs are obtained by diagonalization, and may have parameters that are different than the ones attained by random functions. In particular, while we expect functions hard for (say) linear space on all inputs to exist in cs-$\mathcal{NP}$ (since these follow from $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$), we do not expect such functions to exist in $\mathcal{FP}$ or in $\mathcal{FNP}$. See discussion in Section 2.2.

notions of hardness (e.g., one-way functions, or circuit lower bounds). However, one obstacle for deducing such conclusions is the possibility that $\mathcal{P} = \mathcal{PSPACE}$. To see this, note that a natural attempt at defining a hard problem is to consider the search problem "find an input x on which a given linear-space machine correctly computes f" (since the assumption is that no efficient algorithm can do so). Unfortunately, the foregoing problem is only non-trivial if such inputs *exist* (i.e., f must be computable in linear space on *some* inputs for the problem to be non-trivial). If f is hard on all inputs, then the search problem is trivial; and in particular, if $\mathcal{P} = \mathcal{PSPACE}$ then there is indeed a function in $\mathcal{FP}$ that is hard for linear space on all inputs.

Additional directions for studying the assumption. Note that if $\mathcal{NP} \neq \mathcal{PSPACE}$, then the hardness assumption in Theorem 1.3 yields a problem in $\mathcal{P}^{\mathcal{NP}}$ that is hard for fixed probabilistic exponential time (i.e., there is $c > 1$ such that the problem is hard for probabilistic time 2^{n^c}); specifically, this problem is obtained by combining the cs-$\mathcal{NP}$ verifier for TQBF deduced in Theorem 1.3 with the refuter of Theorem 1.6. We pose the question of whether one can deduce stronger notions of hardness by combining our assumption with $\mathcal{P} \neq \mathcal{PSPACE}$ or with $\mathcal{NP} \neq \mathcal{PSPACE}$. Another useful direction would be finding consequences of Assumption 1.2 in cryptography or in complexity other than non-interactive simulations of proof systems.

2 Technical overview

Many of the proofs in this paper are remarkably simple. We start with the simplest one, which is the construction of zaps and NIWIs in the uniform model (i.e., Theorem 1.8). Then we describe the simulation of $\mathcal{PSPACE}$ in cs-$\mathcal{NP}$ from a sufficient and necessary assumption. Finally, we explain the more technically involved parts of this work, which prove that targeted HSGs suffice for non-black-box derandomization (equivalently, list-CIHFs suffice for Fiat-Shamir) of certain proof systems with super-constantly many rounds, and also suffice for obtaining NIZK.

2.1 Non-interactive zero-knowledge

The gist of the argument is as follows. The hard function f assumed in Theorem 1.8 can be thought of as a keyless, uniformly secure CIHF for a certain class of searchable relations (as defined in [CCH+19]). Such an f suffices for applying Fiat-Shamir to standard zero-knowledge protocols for $\mathcal{NP}$, and it yields a zap/NIWI (and a NIZK, using a uniform version of the FLS trick [FLS99]). The only complications, which are minor, arise from working in the uniform setting.

Let us explain this more slowly. Consider Blum's [Blu86] classical zero-knowledge protocol for the $\mathcal{NP}$-complete problem of Hamiltonicity. On input graph G, the honest prover chooses a random permutation π and sends a commitment to π and to the graph $\pi(G)$. The verifier tosses a coin b. If $b = 0$ the verifier asks the prover to decommit entirely (and thus catches the prover in case the graph to which it committed is not $\pi(G)$), and if $b = 1$ it asks the prover to decommit to a Hamiltonian cycle in the graph sent (and thus catches the prover in case the graph is $\pi(G)$ but G is not Hamiltonian). Indeed, the verifier's view can be easily simulated, since when $b = 0$ it receives a random isomorphic copy of G, and when $b = 1$ it just receives a random cycle graph. This base protocol is repeated in parallel, to obtain negligible soundness error.[12]

[12] Parallel repetition of this protocol is unlikely to preserve zero-knowledge (see [CCH+19; HLR21]), but it does preserve WI (and honest-verifier zero-knowledge), which suffice for our target applications.

As observed by Canetti *et al.* [CCH+19], in the base protocol, for every non-Hamiltonian graph G and every commitment sent by the prover, there is *at most one* verifier challenge that still allows the prover to convince the verifier. (If the prover committed to a permutation π but to a graph that is not $\pi(G)$, then with challenge $b = 0$ the verifier will reject no matter how the prover responds; and if the prover committed to a permutation π and to $\pi(G)$, then $\pi(G)$ is not Hamiltonian, so choosing $b = 1$ will cause the verifier to reject.) This fact relies on using a statistically binding commitment, and remains true when repeating the protocol in parallel.

Thus, given G and commitments $\vec{c} = (c_1, \ldots, c_t)$ (for a t-fold parallel repetition), there is at most one bad challenge $B(G, \vec{c})$ that allows to mislead the verifier. Moreover, the mapping of $(G, \vec{c})$ to $B(G, \vec{c})$ can be computed by breaking the commitment scheme (to see if the graph sent is $\pi(G)$), and this can be done using an $\mathcal{NTIME}[n^{\Omega(1)}]$ oracle. (See further discussion of the parameters in Section 2.2.)

Now, consider a verifier that chooses its challenge deterministically by computing $f(G, \vec{c})$. The key observation is that *if $f(G, \vec{c}) = B(G, \vec{c})$ (i.e., if the function "hits" the bad challenge), then the algorithm for B computes f at input $(G, \vec{c})$*. However, we assumed that f is hard to compute with an $\mathcal{NTIME}[n^{\Omega(1)}]$ oracle over all efficiently samplable distributions! Hence, no malicious adversary can find input G and commitments $\vec{c}$ such that $f(G, \vec{c})$ is a bad challenge for the verifier, except with negligible probability. This guarantees soundness against uniform adversaries when choosing the challenges deterministically. At this point, instead of a deterministic protocol, the prover can just send in advance $\vec{c}$, the challenge $f(G, \vec{c})$, and its final response.

A minor complication in the uniform setting. Recall that we repeat the base protocol in parallel, say for $t = n^{\Theta(1)}$ times (to achieve negligible soundness error). The standard security reduction for proving zero knowledge (or WI) uses an adversary that distinguishes the simulator from the actual protocol in order to break the commitment scheme, and the underlying proof of that relies on a standard hybrid argument. However, hybrid arguments do not work as-is in the uniform setting, since we cannot just non-uniformly hard-wire values for the "non-critical" blocks in a way that maintains the advantage (for a closely related open problem, see the question of "computational mergers" in [CT21]). In our case, computing certain blocks of the hybrid distribution requires access to the witness w (i.e., to a Hamiltonian cycle in G).

Fortunately, we can bypass this obstacle, because we are only interested in *uniformly secure zero-knowledge*, where the simulator is indistinguishable from the actual protocol only over efficiently generateable pairs (G, w) (see Definitions 3.3 to 3.5). Thus, in our security reduction we start from an adversary that produces both G and a witness w, and we can use w to efficiently generate the needed blocks in the hybrid distribution. Details appear in Section 5.1.

2.2 Simulating $\mathcal{PSPACE}$ in cs-$\mathcal{NP}$

Let us now explain why the hardness assumption in Theorem 1.3 is sufficient and necessary for $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$. As mentioned in Section 1.2, our proof strategy for simulating $\mathcal{PSPACE}$ in cs-$\mathcal{NP}$ is to "derandomize $\mathcal{IP} = \mathcal{PSPACE}$", with computational soundness; that is, we simulate the underlying interactive proof system by a cs-$\mathcal{NP}$ protocol.

Specifically, we will show that assumptions as in Theorems 1.3 and 1.7 suffice to simulate a suitable version of the interactive proof system of Goldwasser, Kalai, and Rothblum [GKR15].[13]

[13] As noted already in their original work, their proof system can indeed be used to decide $\mathcal{PSPACE}$, and this is

As in Section 2.1, our high-level approach will be to define a notion of bad verifier challenges, and argue that the set of bad challenges at each round is a singleton that can be computed relatively efficiently (i.e., by a low-space machine). At each round, the derandomized protocol applies f to the transcript up to that round, instead of having the verifier choose a challenge randomly. By our assumption, no efficient adversary can find a transcript Π (i.e., an input and prover responses) such that $f(\Pi)$ equals the single bad challenge, except with negligible probability.

The notion of a bad challenge in this context follows from the observation of Canetti *et al.* [CCH+19] that the proof system of [GKR15] has round-by-round soundness. We will elaborate on this notion below when we define a subclass of such systems (see Section 2.3.2), but the crucial point for now is that on input $x \notin L$, in each round i the following holds. There is a set B_i of challenges of small density $\epsilon > 0$ (i.e., the bad ones) such that, if the verifier accidentially chooses a challenge in B_i, then the prover can mislead it into thinking that $x \in L$; but if the verifier avoids B_i, then in the next round the challenges are again partitioned into bad ones B_{i+1} and non-bad ones, where the density of B_{i+1} is exactly the same, i.e. $\epsilon > 0$. (And in the last round, if the verifier consistently avoided bad challenges, it rejects.) In other words, as long as the verifier avoids a set of small fixed density $\epsilon > 0$ at each round, the prover cannot mislead it.

The missing piece is arguing that a suitable version of the protocol of [GKR15] has one bad challenge in each round (i.e., $|B_i| = 1$). A version with a sufficiently close property was presented by Kalai, Lombardi, and Vaikunatanatan [KLV23], following previous variations in [Gol18; KPY19; JKK+21]: they showed a protocol where the number of bad challenges at each round is constant (i.e., $|B_i| = O(1)$). Accordingly, we will argue that there are $O(1)$ low-space machines, each computing one of the bad challenges. If there is an adversary (i.e., a malicious prover) that samples Π such that the resulting protocol is insecure, with noticeable probability, on infinitely many input lengths, then at least one of these machines will succeed in computing $f(\Pi)$ with noticeable probability on infinitely many input lengths. This contradicts the hardness of f.

We defer the full description of the proof system itself (i.e., of the version with $O(1)$ bad challenges at each round) to Section 4, since it relies on many details of the proof system of [GKR15], and since the key observations already appeared in previous work (i.e., [KLV23]).

As for the other direction, assuming $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$ we want a function that is hard for (say) linear space on all efficiently samplable distributions. To do so, we first construct a function in $\mathcal{PSPACE}$ with such hardness, by straightforward diagonalization; in fact, we construct a function that is hard for each linear-space machine on *all inputs* (except at most finitely many). Using the hypothesis on each output bit, this function is also in $\texttt{cs-}\mathcal{NP}$.

On the parameters of the two hard functions. In the proof above, in one direction we relied on a function in $\texttt{cs-}\mathcal{NP}$ that is hard for small space on all efficiently samplable distributions, and in the reverse direction we deduced a function in $\texttt{cs-}\mathcal{NP}$ that is hard for small space on all inputs. (Indeed, the former and more relaxed assumption suffices for $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$.)

We stress, though, that the two aforementioned hard functions may be quite different in some of their parameters. In particular, the function that we use to deduce that $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$ must shrink its input, i.e. it maps $N = |\Pi|$ bits to at most N^ϵ bits, similarly to a hash function.[14] This seeming technicality is important: At the technical level, the proof does not carry through

true also for the modified version we use.

[14] Another difference is the running time of the efficient $\texttt{cs-}\mathcal{NTIME}$ prover (and corresponding adversaries) for the hard function; see discussion after Theorem 4.3.

with a function that is not shrinking (and neither does the zap/NIWI construction); and a more fundamental reason for this is that under plausible assumptions, a proof that carries through with a non-shrinking f would imply that $\mathcal{PSPACE} = \mathcal{NP}$, rather than just $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$.[15] Similarly, our alternative proofs that rely on targeted HSGs (equivalently, on list-CIHFs) which are constructed from hard functions also crucially need a hard function that is shrinking, to obtain a sufficiently short seed / small list. (Further details appear in the next section.)

2.3 Targeted HSGs/list-CIHFs suffice for all of our results

Recall that CIHFs output a string that, with high probability, avoids any sparse relation from a certain class of relations. In the relaxation of list-CIHFs, we allow the function to output a list, and require that at least one element in the list avoids the sparse relation (see Definition 6.6).

The proofs described so far, in Sections 2.1 and 2.2, crucially rely on a function f that has a single output. However, as mentioned in Section 1.4, all results in the current paper also hold when using a function that outputs a list of candidates (i.e., a targeted HSG / list-CIHF). Specifically, we can tolerate a list of size n^ϵ, when $\epsilon > 0$ is a sufficiently small constant.[16] For $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$ we require that at least one element will "hit" any dense set recognizable in small space (see Theorem 7.4), and to get NIWI/zaps/NIZK for $\mathcal{NP}$ the dense set need only be recognizable in linear time with an $\mathcal{NTIME}[n^{\Omega(1)}]$ oracle (see Theorem 7.8).

The fact that list-CIHFs suffice for all of our results is surprising, and the proofs of this fact are very different than the simple ones above. As one illustration for the challenge, recall that when derandomizing proof systems with R rounds, if we use a pseudorandom list of L verifier challenges in each round, the total number of challenge-sequences is L^R; thus, when $R = \text{poly}(n)$ we get exponentially many challenge-sequences (even with $L = 2$).[17] This is usually perceived as a "dead end" for derandomizing such systems, and it is the reason that derandomization traditionally focused on systems with constantly many rounds. The key point in this context is that this obstacle can be overcome for certain proof systems, even ones with polynomially many rounds, when aiming only for *computational soundness*.

2.3.1 Zero-knowledge from list-CIHFs

Let us consider Blum's Hamiltonicity protocol again, as in Section 2.1. Recall that on input graph G, the prover commits to a permutation π and to a graph $\pi(G)$, the verifier sends a bit $b \in \{0,1\}$, and the following holds: If $b = 0$ the prover decommits to $(\pi, \pi(G))$, and otherwise the prover

[15] To see this, recall that a random function mapping N bits to N^ϵ bits is hard for bounded-space machines on all efficiently samplable distributions; even if such function exists in $\mathcal{FP}$ or in $\mathcal{FNP}$, it still yields the same conclusion $\mathcal{PSPACE} \subseteq \texttt{cs-}\mathcal{NP}$. However, a random function mapping N bits to $N + O(1)$ bits is hard for linear space on *all inputs*. Assuming that such a function f exists in $\mathcal{FP}$ or $\mathcal{FNP}$, if the proof above would work with f, we could use this function and deduce soundness for all transcripts (yielding an $\mathcal{NP}$ system rather than a $\texttt{cs-}\mathcal{NP}$ system).

[16] Similarly to Section 2.2, this does not seem to be a coincidence or an artifact of techniques. Recall that a random non-shrinking function $f\colon \{0,1\}^N \to \{0,1\}^{N+O(1)}$ is hard for linear space on all inputs (except finitely many). If such f exists in $\mathcal{FP}$, we can use it to construct a targeted HSG that is secure on all but finitely many inputs, albeit with large list size $\text{poly}(N)$. If our proof would support such list size, then we could use the targeted HSG to deduce $\mathcal{NP} = \mathcal{PSPACE}$. In contrast, our proof supports list size N^ϵ, which we only know how to obtain from shrinking functions $N \mapsto N^{\Theta(\epsilon)}$ (and a random shrinking function does not seem to be hard on all but finitely many inputs).

[17] An alternative and more standard view is that the seed length is additive across rounds, so with seed length $\ell = \log(L)$ per round, the total seed length is $R \cdot \ell$. Enumerating would require time $2^{R \cdot \ell}$.

decommits to a Hamiltonian cycle in $\pi(G)$. When performing a t-fold parallel repetition of the protocol, the prover chooses t permutations and commits accordingly, the verifier sends a sequence $\vec{b} \in \{0,1\}^t$, and the prover responds with t decommitments.

Now, assume that $\vec{b}$ is not chosen at random, but using a list-CIHF, denoted f. That is, on input G and commitments $\vec{c}$, the prover and verifier both compute $f(G, \vec{c})$ to obtain a list $\vec{b}^1, ..., \vec{b}^L$ of possible challenges, and the prover responds with decommitments to each of the L challenges. Working with a statistically binding commitment as in Section 2.1, and assuming that f avoids a suitable relation (see next), this protocol is sound. But is it zero-knowledge, or at least WI?

Unfortunately, the construction fails miserably on that front. For example, when the first bit in $\vec{b}^1$ is 0 and the first bit in $\vec{b}^2$ is 1, the prover has to simply reveal a Hamiltonian cycle in G.

This problem is somewhat reminiscent of *resettable zero-knowledge* [CGG+00]. Indeed, similarly to our situation, in resettable zero-knowledge the verifier is allowed to continue the execution of the protocol using different choices. Unfortunately, constructions of resettable zero-knowledge use private randomness, which we do not know how to derandomize. Fortunately, resettable zero-knowledge seem like an overkill, since it allows the verifier to continue the execution of the prover for any polynomial number of times (where the polynomial is not a priori bounded). This is in contrast to our setting, in which the number of resets is relatively small (i.e., L).

Zero-knowledge PCPs to the rescue. The key observation is that, when working with an abstraction of the above protocol – namely, instantiating the "commit-and-open" approach with zero-knowledge PCPs (zkPCP), introduced by Kilian, Petrank, and Tardos [KPT97], as is implicit in [IKO+09] – we can ensure that the protocol remains zero-knowledge even when the verifier sends more queries. This is since zkPCPs are purposefully designed to be zero-knowledge for a malicious verifier that sends more queries than the honest verifier.

While there are several variants of zkPCPs, we focus on the relatively simple notion of *non-adaptive* zero-knowledge, which means that for every sufficiently small, but otherwise arbitrary, set of queries Q, the restriction of the PCP to coordinates in Q can be generated by a ppt simulator which only sees the main input. Specifically, our construction uses the following ingredients:

- A statistically binding (and computationally hiding) non-interactive commitment scheme. For now, let us assume for simplicity that the commitment scheme is in the plain model.
- A q-query zkPCP that is resilient (i.e., zero-knowledge) for up to $q_{\max} \gg q$ queries. Denote the length of the PCP by ℓ.
- A targeted HSG G that gets as input an instance x and commitments $c_1, \ldots, c_\ell$ to ℓ PCP symbols, and prints a list S of candidate random strings for the zkPCP verifier. We set up the parameters so that $q_{\max} \geq |S| \cdot q$.

In the "commit-and-open" protocol, on input x the prover sends a commitment $\vec{c}$ to the symbols of a zkPCP witness for x, the verifier sends challenge queries Q, and the prover decommits to the locations specified in Q. In our derandomized version, the prover and verifier compute the targeted HSG $G(x, \vec{c})$, which yields a set S of random strings for the verifier; each $\rho \in S$ specifies queries Q_ρ, and the prover answers all queries in $Q = \cup_{\rho \in S} Q_\rho$.

Completeness and soundness rely on the perfect completeness of the underlying commitment and zkPCP procotols and on the fact that the commitment is statistically binding. Specifically, the

key point for soundness is that an adversary with an $\mathcal{NP}$ oracle, trying to distinguish the outputs of the targeted HSG from random strings, can break the commitments $\vec{c}$ and reveal the zkPCP witness. Therefore, if x is a "no" input but $G(x, \vec{c})$ chooses queries Q that all avoid the dense set of bad queries, the adversary can recognize that and distinguish these queries from random ones. (Recall that G is guaranteed to be pseudorandom for such adversaries over inputs $(x, \vec{c})$ coming from any polynomial-time samplable distribution; thus, no uniform ppt adversary can find x and $\vec{c}$ that allow misleading the derandomized verifier, except with negligible probability.)

Let us now turn to prove witness-indistinguishability (recall that we're aiming for a zap/NIWI, and a NIZK will follow using the FLS trick [FLS99]). Intuitively at least, since the prover reveals at most $|S| \cdot q$ coordinates of the PCP, as long as the zkPCP is resilient to this number of queries (i.e., $q_{\max} > |S| \cdot q$) we should be in good shape. Examining things in more detail however, we observe that the choice Q of coordinates to be opened depends on the commitments themselves. This well-known problem (usually called "selective opening") can be handled with known techniques, and yet these techniques force us to instantiate our protocol with strong ingredients (i.e., strong zkPCP and commitment scheme; see below). Specifically, to handle the problem we construct a simulator that guesses in advance a similarly sized set Q_{sim} and generates a PCP that is correct on these coordinates (using the zkPCP simulator). In case the simulator's prediction is correct (i.e., the set of queries Q generated via the targeted PRG is equal to Q_{sim}) then it can complete the simulation, and it is not hard to show that it is indeed correct with probability roughly $\ell^{-|Q|}$.[18] Thus, we can repeat the entire process $\ell^{O(|Q|)}$ times so that the simulator guesses correctly in one of the iterations with high probability, in which case we can finish the simulation.

An improved zkPCP, and the required commitment scheme. The analysis above crucially relies on a zkPCP with *perfect zero-knowledge* (or very close to that).[19] Most known zkPCPs (including the original construction in [KPT97]) do not offer perfect security, and so we cannot use them in our analysis. Instead, our starting point is a recent *perfect* zkPCP construction due to Hazay *et al.* [HVW22] (improving upon a zkPCP implicit in [IKO+09]). The randomness complexity of the construction of [HVW22] is too high for our purposes, and so we present a derandomized construction, which uses an optimal number of random coins (one reason that we are able to do so is that in our setting we can settle for constant soundness error, in contrast to [HVW22]). A result statement and technical details appear in Theorem 3.10 and Appendix A.

In addition, the super-polynomial number of iterations of the simulator means that the commitment scheme has to be resilient against adversaries with super-polynomial running time (roughly speaking, because the security reduction needs to run the simulator). This is the reason for our assumption that the commitment scheme has sub-exponential security.

[18] Loosely speaking, the choice of the set Q_{sim} depends only on the "encrypted" PCP string. The hiding property of the commitment lets us then argue that the distribution of Q does not depend (noticeably) on Q_{sim}. Thus, we can think of Q_{sim} as being chosen at random *after* Q is generated.

[19] In a nutshell, this is because the security reduction introduces an $\ell^{|O(|Q|)}$ multiplicative loss in the the advantage of a potential adversary for the system (roughly speaking, this is due to a hybrid argument over the $\ell^{O(|Q|)}$ iterations of the simulator). This means that we cannot afford an additive loss of even $2^{-|Q|}$ when switching between a real vs. simulated PCP.

2.3.2 Derandomizing [GKR15], and batchable proof systems

The proof system of [GKR15] has $R = \text{poly}(n)$ rounds. Recall that if we use a targeted HSG with L challenges in each round, then the standard approach of enumerating over all pseudorandom choices yields an exponential running time (i.e., at least L^R).

However, let us examine the enumeration construction from a different perspective. This construction can be viewed as specifying a deterministic interactive protocol with R rounds, in which the verifier uses all of the pseudorandom challenges in each round. In the first round, the verifier prepares L challenges $r_1^{(1)}, ..., r_1^{(L)}$ and sends them to the prover; then, for each response $\pi_1^{(i)}$ from the prover, the verifier and the prover "branch" to execute an independent continuation of the protocol with partial transcript $\left\langle x, r_1^{(i)}, \pi_1^{(i)} \right\rangle$. Indeed, the number of branches (i.e., number of leaves in the corresponding tree) grows exponentially in the number of rounds.

Can we get a proof system that supports verifying several challenges in each round "in a batch", without branching exponentially across rounds? We show that the proof system of [GKR15] can indeed be modified to support this, and prove that in general, any system that supports this can be derandomized (with computational soundness) using a targeted HSG.

At a high level, we say that a proof system is L-batchable if it allows the verifier to send L challenges in each round *without branching into L independent executions* in subsequent rounds, and without (prohibitively) increasing the verification time. As far as we are aware, this notion was not introduced formally before, but it is implicit in prior works (e.g., in [GKR15; CHK+19; RRR21], and see the related [ACY22]). We first demonstrate our idea by explaining how the sumcheck protocol can be modified to handle several challenges in each round without branching, using the well-known "2-to-1" idea of [GKR15]. Then we describe the general notion of batchable proof systems, and explain why targeted HSGs suffice for derandomizing such systems.

A simple illustration: The sumcheck protocol. Consider the sumcheck protocol of Lund *et al.* [LFK+92] for the claim $\sum_{x_1,...,x_n \in \{0,1\}} P(x_1, ..., x_n) = K$, where $P \colon \mathbb{F}^n \to \mathbb{F}$ is a low-degree polynomial and $K \in \mathbb{N}$. Think of an $|\mathbb{F}|$-ary tree in which the root is labeled by $\sum_{x_1,...,x_n \in \{0,1\}} P(x_1, ..., x_n)$, each edge corresponds to a field element, and each node reached by path $a_1, ..., a_i \in \mathbb{F}$ is labeled by $\sum_{x_{i+1},...,x_n \in \{0,1\}} P(a_1, ..., a_i, x_{i+1}, ..., x_n)$. At round i, the previous verifier challenges $a_1, ..., a_{i-1} \in \mathbb{F}$ specify a node, and the prover sends a univariate $q_i \colon \mathbb{F} \to \mathbb{F}$ that represents the labels of the $|\mathbb{F}|$ children of this node; that is, q_i is supposed to be the univariate $p_{a_1,...,a_{i-1}}(r) = \sum_{x_{i+1},...,x_n \in \{0,1\}} P(a_1, ..., a_{i-1}, r, x_{i+1}, ..., x_n)$. The verifier checks that $q_{i-1}(a_{i-1}) = q_i(0) + q_i(1)$ (i.e., that the claimed label of the node $a_1, ..., a_{i-1}$ matches the claimed labels of its children corresponding to 0 and to 1), chooses a random edge $a_i \in \mathbb{F}$, and continues to the next round.

Now assume that in each round the verifier sends L challenges instead of a single challenge. We enter round i with each previous challenge specifying a node at distance $i-1$ from the root; that is, the L challenges specify the nodes $\left\{ \vec{a}_{i-1}^{(t)} = (a_1^{(t)}, ..., a_{i-1}^{(t)}) \right\}_{t \in [L]}$, and the verifier wants to verify the labels claimed by q_{i-1} to all of these nodes. Instead of branching to L independent executions in round i, the verifier and prover interpolate a manifold $C^{(i)} \colon \mathbb{F} \to \mathbb{F}^i$ of degree $L-1$ that passes through the $2L$ children of these L nodes corresponding to edges 0 and 1. Specifically, denoting the first $2L$ elements in $\mathbb{F}$ by $\{\sigma_{t,b}\}_{t \in [L], b \in \{0,1\}}$, we define $C^{(i)}(\sigma_{t,b}) = (a_1^{(t)}, ..., a_{i-1}^{(t)}, b)$, and the verifier expects the prover to send the univariate

$p'_{\vec{a_{i-1}}^{(1)},\ldots,\vec{a_{i-1}^{(L)}}}(\sigma) = \sum_{x_{i+1},\ldots,x_n \in \{0,1\}} P(C^{(i)}(\sigma), x_{i+1}, \ldots, x_n)$.

The point is that if L is small, then $p'_{\vec{a_{i-1}}^{(1)},\ldots,\vec{a_{i-1}^{(L)}}}$ is still of low degree (i.e., it is of individual degree $\deg(P) \cdot (2L-1)$), and thus the standard analysis of the sumcheck protocol can be replicated. Specifically, the verifier first checkes that the labels claimed by q_{i-1} to the L nodes match the labels claimed by q_i to their children, then chooses L random challenges $\sigma_i^{(1)}, \ldots, \sigma_i^{(L)} \in \mathbb{F}$, and continues to the next round with the L nodes $\left\{\vec{a}_i^{(t)} = C^{(i)}(t)\right\}_{t\in[L]}$ at distance i from the root.[20]

If the label claimed by q_i to some node $(a_1^{(t)}, \ldots, a_{i-1}^{(t)}, b)$ is false, then with high probability the prover will be forced to send a q_{i+1} that also claims a false label for some node (since q_i is low degree and the verifier will perform subsequent consistency checks of q_i with q_{i+1}).

Round-by-round soundness. To set up the stage for the definition of batchable proof systems, let us first consider proof systems with round-by-round soundness, as defined by Canetti *et al.* [CCH+19]. Loosely speaking, in such systems, the challenge-response interaction at each round can be thought of as a game with a binary "yes/no" outcome. That is, the probabilities of the game can only be of *two predetermined types*, depending on the transcript entering the round: Either probability 1 for "yes", or probability at least $1-\epsilon$ for "no", where $\epsilon = \epsilon(n) > 0$ is predetermined (i.e., it does not depend on the previous interaction). More accurately:

1. There are notions of "good" partial transcripts and of "doomed" partial transcripts, where in particular $x \in L$ is a good initial transcript and $x \notin L$ is a doomed one.
2. At each round i, if the partial transcript $\pi_{\leq i-1}$ entering the round is doomed, then with probability $1-\epsilon$ over the verifier's challenge, any response by the prover will yield a transcript that is still doomed. On the other hand, if $\pi_{\leq i-1}$ is good, then the honest prover can always respond in a way that preserves goodness.
3. When the interaction concludes, the verifier accepts good transcripts, and with probability $1-\epsilon$ rejects doomed transcripts.

The crucial point is that the probability bound of $1-\epsilon$ is fixed in advance, for all rounds, as long as the transcript entering the round is either good or doomed. For details, see Definition 7.1.

Batchable proof systems. Batchable proof systems are a special case of proof systems with round-by-round soundness. A proof system is L-batchable if at each round, the verifier can send L challenges and receive L responses (instead of just a single challenge and response), and we think of each round as the AND of L games. Specifically:

1. At each round i, we conduct a mental experiment of what would happen if the verifier would send just a single challenge, answered with a single response. We extend the notions of good and doomed partial transcripts to include transcripts in which L challenge-response pairs were exchanged in previous rounds, whereas in the last (i.e., current) round only a single challenge-response pair was exchanged.

[20] In more detail, the consistency check between q_{i-1} and q_i is as follows. Denoting the challenges sent in the previous round by $\sigma_{i-1}^{(1)}, \ldots, \sigma_{i-1}^{(L)} \in \mathbb{F}$, the verifier checks that $q_{i-1}(\sigma_{i-1}^{(t)}) = q_i(\sigma_{t,0}) + q_i(\sigma_{t,1})$ for all $t \in [L]$.

2. Fix a partial transcript $\pi_{\leq i-1}$ going into round i. We say that round i is a "yes" round if for any (single) challenge sent by the verifier, the prover has a response such that the transcript will be good. We say that this is a "no" round if with probability $1-\epsilon$ over a (single) challenge by the verifier, any response will yield a doomed transcript.

3. If we enter round i when *at least one* of the L challenge-response pairs in round $i-1$ yielded a doomed transcript,[21] then round i is a "no" round. If all L challenge-response pairs in round i yielded a good transcript, then round i is a "yes" round.

The point is that exchanging L challenge-response pairs in round $i-1$ still allows round i to be conducted similarly to the case of round-by-round soundness (i.e., with a single challenge-response pair, rather than branching into L games that correspond to the previous L responses). And again, every round will be either a "yes" round or a "no" round (in particular, with a fixed and predetermined probability bound of $1-\epsilon$), as long as the following invariant is maintained: Either all challenge-response pairs sent in the last round yielded a good transcript, or at least one of them yielded a doomed transcript. For the formal definition, see Definition 7.2.

The modified version of sumcheck above (i.e., with the manifolds $C^{(i)}$) is a $\mathrm{poly}(n)$-batchable proof system for #SAT with a polynomial-time verifier. More generally, applying the same idea albeit in a more involved way, the proof system of [GKR15] can be transformed into a $\mathrm{poly}(n)$-batchable proof system for TQBF with a polynomial-time verifier, and to an $n^{o(1)}$-batchable proof system for logspace-uniform $\mathcal{NC}$ with a near-linear time verifier. The precise details are quite cumbersome (with subtle parameters), and therefore we defer explanations to Section 7.1.4.

Targeted HSGs suffice for derandomization of batchable proof systems. To wrap things up, note that a targeted HSG with list size L that is secure over all efficiently samplable distributions indeed suffices to derandomize L-batchable proof systems with computational soundness.

Specifically, consider a deterministic choice of L challenges in each round i that is obtained by applying the targeted HSG to the partial transcript $\pi_{\leq i-1}$. Since the targeted HSG is pseudorandom on all efficiently samplable distributions, we can assume that it is pseudorandom on input $\pi_{\leq i-1}$ arising from the choice of responses by the efficient uniform prover (with all but negligible probability). Now, if $\pi_{\leq i-1}$ is such that round i is a "no" round, then a random choice of a single challenge by the verifier in round i yields a doomed transcript. By the pseudorandomness of the targeted HSG on input $\pi_{\leq i-1}$, at least one of the L challenges will yield a transcript $\pi_{\leq i}$ that makes i a "no" round too (no matter how the prover responds). In other words, when choosing the challenges using the targeted HSG, with all but negligible probability over the prover's responses, if $x \notin L$ then all rounds are "no" rounds and the verifier rejects.

For more details, including the precise class of algorithms that the targeted HSG needs to be pseudorandom for in order to derandomize the batchable proof system, see Section 7.1.3.

3 Preliminaries

The machine model throughout this paper is the RAM model, and the specific choice is not crucial for our results. We say that a function $\epsilon : \mathbb{N} \to [0,1]$ is negligible if for every polynomial p there exists n_0 such that $\epsilon(n) < 1/p(n)$, for every $n > n_0$.

[21] That is, when considering the partial transcript up to round $i-1$, and in round $i-1$ including only that single challenge and response, the resulting partial transcript is doomed.

Let us recall the standard definition of a distinguisher for a distribution:

Definition 3.1 (distinguishers)**.** *We say that* $D\colon \{0,1\}^n \to \{0,1\}$ *is an* ϵ*-distinguisher for a distribution* $\mathbf{w}_n$ *over* n*-bit strings if*

$$\left| \Pr_{r\sim \mathbf{w}_n}[D(r)=1] - \Pr_{r\sim \mathbf{u}_n}[D(r)=1] \right| \geq \epsilon\,.$$

We say that D *is an* ϵ*-distinguisher for a multiset* $S \subseteq \{0,1\}^n$ *if* D *is an* ϵ*-distinguisher for the uniform distribution over* S*.*

In our results, the distinguisher $D = D_x$ will frequently be modeled as a uniform machine M with access to an auxiliary input x; that is, $D_x(r) = M(x,r)$.

3.1 Non-interactive zero-knowledge in the uniform setting

In this section we define variants of non-interactive zero-knowledge proofs that are secure against uniform adversaries: NIZK, zaps, and NIWIs. Towards doing so, we first define non-interactive arguments for $\mathcal{NP}$-relations in the CRS model, extending the definition of cs-$\mathcal{NTIME}$.

Non-interactive arguments for $\mathcal{NP}$-relations in the CRS model. We extend the definition of cs-$\mathcal{NTIME}$ (i.e., Definition 1.1) in two ways. First, in this section we are interested in constructing argument systems for non-deterministic computation. In this context, it is natural to give the honest prover access to the $\mathcal{NP}$ witness. Second, we further extend the definition by giving both parties access to a common random string (CRS).

For a relation R, we denote by $L(R) = \{x : \exists w, (x,w) \in R\}$.

Definition 3.2 (non-interactive argument)**.** *A non-interactive argument system in the CRS model for an* $\mathcal{NP}$ *relation* R *is a pair of algorithms* (P,V)*, where* P *is ppt and* V *is deterministic polynomial-time, that satisfy the following requirements:*

- **(Perfect completeness:)** *For every* $(x,w) \in R$ *it holds that*

$$\Pr_{\substack{crs\leftarrow\{0,1\}^{|x|}\\ \pi\leftarrow P(crs,x,w)}}[V(crs,x,\pi)=1] = 1.$$

- **(Adaptive soundness:)** *For every ppt algorithm* $\widetilde{P}$ *it holds that*

$$\Pr_{\substack{crs\leftarrow\{0,1\}^n\\ (x,\pi)\leftarrow\widetilde{P}(crs)}}\left[\big(x \notin L(R)\big) \text{ and } \big(V(crs,x,\pi)=1\big)\right]$$

  is a negligible function (in n*).*

A non-interactive argument system is said to be in the *plain model* if the *crs* is omitted in the above definition. We note that non-interactive arguments in the plain model are almost identical to the notion of cs-$\mathcal{NTIME}[T_V, T_P]$ (with T_V and T_P being arbitrary polynomials). The key difference is that here we give the honest prover access to the witness.

3.1.1 Non-interactive zero-knowledge

We next define uniform non-interactive zero-knowledge (NIZK). The textbook definition of zero-knowledge usually refers to non-uniform adversaries - that is, both soundness and zero-knowledge hold wrt polynomial-size circuits. In contrast, our derandomization technique yields soundness against *uniform* algorithms (i.e., bounded-time Turing machines). For symmetry we also define the zero-knowledge property wrt uniform algorithms, but remark that this is not inherent to our technique; we could also achieve non-uniform zero-knowledge if we rely on cryptographic primitives (namely, one-way functions) that have non-uniform security.

Our definitions for uniform security generally follow those in [BLV06], which refine prior definitions due to Goldreich [Gol93].

Definition 3.3 (NIZK). *A non-interactive zero-knowledge (NIZK) argument system in the CRS model for an $\mathcal{NP}$ relation R is a non-interactive argument system (P, V) for R in the CRS model that also satisfies the following condition. There exists a pair of ppt algorithms (S_1, S_2) such that for every pair of ppt algorithms (A_1, A_2) it holds that:*

$$\left| \Pr_{\substack{crs\in\{0,1\}^n \\ (x,w,z)\leftarrow A_1(crs) \\ \pi\leftarrow P(crs,x,w)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w)\in R \end{bmatrix} - \Pr_{\substack{(crs,\tau)\in S_1(1^n) \\ (x,w,z)\leftarrow A_1(crs) \\ \pi\leftarrow S_2(x,\tau)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w)\in R \end{bmatrix} \right|$$

is a negligible function (in n).

Throughout this work, unless stated otherwise, whenever we say NIZK we mean NIZK argument system (the same is also applicable to zaps and NIWI that are defined below).

Note that the simulator in Definition 3.3 is split into two parts. The first part S_1 generates a simulated CRS and potentially a corresponding trapdoor. The second part, which is given the input x and the trapdoor needs to generate a simulated proof. We require zero-knowledge also against adaptive adversaries. Namely, we require that any 2-part ppt adversary that first, given the CRS, generates an instance/witness pair (as well as some potential auxiliary input z) and later tries to distinguish a real vs. simulated proof, can only do so with a negligible advantage.

Discussion: Definitional choices for NIZK in the uniform setting. To the best of our knowledge, *non-interactive* zero-knowledge has not been defined previously in the uniform setting (in contrast to uniform interactive zero-knowledge [Gol93; BLV06]) and there are several non-trivial choices that we made in Definition 3.3. We elaborate on these next while noting these issues could have been avoided by using the standard non-uniform definition of NIZK (which our results can achieve if we rely on non-uniformly secure cryptographic primitives).

First, we note that the zero-knowledge property in Definition 3.3 is only required to hold for instance/witness pairs that are generated by an efficient algorithm (namely, A_1). This means for example, that zero-knowledge is not guaranteed when using some "hard-to-find" witness. We find this choice natural and consistent with the typical cryptographic perspective that all parties are efficient, and so the hard-to-find witnesses will simply never appear. This choice is also in line with definitions of adaptive zero-knowledge in the literature, in both the uniform [Gol93] and non-uniform settings [Gol93, Section 4.10.3.2].

Next, we point out two definitional choices that make our notion of zero-knowledge stronger (and hence also make our results stronger), but might seem non-obvious. Specifically:

- In Definition 3.3 we require the existence of a single universal simulator that works for all adversaries, whereas a relaxed definition could require that for every adversary A_1 there exists a corresponding simulator (that works for all distinguishers A_2).
- The instance generator A_1 is allowed to generate some auxiliary information z that is passed on to A_2, but not to the simulator. At first glance it may seem natural that the auxiliary information be provided also to the simulator.

Beyond simply making the notion stronger, the two definitional choices above follow a perspective that is inspired by *interactive* zero-knowledge proof in the uniform setting [Gol93]. In such interactive zero-knowledge proofs the verifier actually consists of three different personas: the first generates the solved instance/witness pairs (this is A_1 in our terminology), the second is V^* which participates in the interaction and tries to extract as much information as possible from the prover, and the third is the distinguisher (i.e., A_2) whose goal is to distinguish the real interaction from the simulated one. In the interactive setting the typical requirement, which we find natural, is that for every V^* there exists a simulator that works for every choice of A_1 and A_2. In the non-interactive case V^* is vacuous (as there is no interaction) and so restricting the definition of the interactive case to this setting means that the simulator is universal.

As for the auxiliary information z generated by A_1. In the interactive setting, this z is passed both to the verifier V^*, the simulator and the distinguisher. However, as highlighted in [Gol01, Section 4.3.3], the definition of computational zero-knowledge allows the auxiliary information to include a portion that is readable only by the distinguisher.[22] In our non-interactive setting, since V^* does not exist, by symmetry there is no need to pass auxiliary information to the simulator. In contrast, and in line with the interactive setting, we do wish to allow auxiliary information to be passed to the distinguisher, and this is the reason for our choice.

3.1.2 Zaps Arguments and NIWIs

Zaps, introduced by Dwork and Naor [DN07], are 2-message public-coin witness indistinguishible (WI) proof systems. Loosely speaking, witness indistinguishibility [FS90] means that the view of the verifier should be computationally indistinguishible no matter which witness the prover uses. We emphasize that the WI property should hold even wrt a malicious ppt verifier. In this work we focus on zap arguments [BFJ+20], in which the soundness condition is computational (in contrast to [DN07] who require statistical soundness).[23] For ease of notation, throughout this work we use the term zap to refer to zap arguments.

In the standard cryptographic setting of *non-uniform security* WI is equivalent to having a computationally unbounded simulator. In contrast, in the uniform setting the connection appears to be only in one direction - unbounded simulation implies witness indistinguishability (see

[22] Technically, this is done using the fact that the distinguisher's running time may be larger than both the verifier and simulator and additional portion is placed in an area of the machine's tape that only the distinguisher can access.

[23] We note that the focus of [BFJ+20] is achieving *statistical* WI, whereas our definition only requires the WI property to be computational.

Lemma 3.6 below).[24] Hence, we define these two notions separately.

Definition 3.4 (unbounded simulator zap)**.** *An* unbounded simulator zap for an $\mathcal{NP}$ relation *R is a non-interactive argument system (P, V) for R in the* CRS *model that also satisfies the following condition. There exists a computationally unbounded algorithm S such that for every pair of ppt algorithms (A_1, A_2) it holds that:*

$$\left| \Pr_{\substack{(crs,x,w,z)\leftarrow A_1(1^n)\\ \pi\leftarrow P(crs,x,w)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w)\in R \end{bmatrix} - \Pr_{\substack{(crs,x,w,z)\leftarrow A_1(1^n)\\ \pi\leftarrow S(crs,x)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w)\in R \end{bmatrix} \right|$$

is a negligible function (in n).

Next, we define the WI variant of zaps (see also the related [BKP+23, Definition 2.16]).

Definition 3.5 (WI zap)**.** *A* WI Zap for an $\mathcal{NP}$ relation *R is a non-interactive argument system (P, V) for R in the* CRS *model, that also satisfies the following condition. For every pair of ppt algorithms (A_1, A_2) it holds that:*

$$\left| \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow P(crs,x,w_1)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w_1),(x,w_2)\in R \end{bmatrix} - \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow P(crs,x,w_2)}} \begin{bmatrix} A_2(crs,x,\pi,z)=1 \\ \wedge \\ (x,w_1),(x,w_2)\in R \end{bmatrix} \right| \tag{3.1}$$

is a negligible function (in n).

A non-interactive witness indistinguishable argument system in the plain model (NIWI) is defined identically to a WI zap except that the CRS is fixed to 1^n. Let us now spell out the argument that an unbounded simulator implies WI for the uniform notion of zaps.

Lemma 3.6. *An unbounded simulator zap is also a WI zap.*

Proof. Let (P, V) be an unbounded simulator zap. We show that it is also a WI zap.

Let S be the unbounded simulator that exists according to Definition 3.4 and let (A_1, A_2) be a pair of ppt algorithms that attempt to violate Definition 3.5. Without loss of generality we assume that with all but negligible probability, A_1 generates (x, w_1, w_2) such that $(x, w_1), (x, w_2) \in R$ since otherwise it can simply abort (without decreasing its advantage in violating Eq. (3.1)).[25]

From Definition 3.4 we have that:

$$\left| \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow P(crs,x,w_1)}} \left[A_2(crs,x,\pi,z)=1\right] - \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow S(crs,x)}} \left[A_2(crs,x,\pi,z)=1\right] \right| \tag{1}$$

[24] The converse direction (witness indistinguishability implies unbounded simulation) is easy to show in the non-uniform setting since the simulator can simply use, say, the lexicographically first witness. This type of argument seems to fail in the uniform setting since there the WI property only holds for "easy to find" witnesses.

[25] In more detail, suppose (A_1, A_2) violate the WI zap condition with advantage $1/p(n)$ for some polynomial p (and infinitely many n). We can now construct A_1' that repeatedly invokes A_1, at most $p(n) \cdot n$ times, or until it generates $(x, w_1), (x, w_2) \in R$ (since $R \in \mathcal{NP}$ it can check this condition). With all but $2^{-\Omega(n)}$ probability, A_1 indeed generates $(x, w_1), (x, w_2) \in R$. Also, observe that conditioned on $(x, w_1), (x, w_2) \in R$, the statistical distance between the distribution generated by A_1 is the same as generated by A_1'.

is negligible. Similarly, we have that

$$\left| \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow P(crs,x,w_2)}} \left[A_2(crs,x,\pi,z)=1\right] - \Pr_{\substack{(crs,x,w_1,w_2,z)\leftarrow A_1(1^n)\\ \pi\leftarrow S(crs,x)}} \left[A_2(crs,x,\pi,z)=1\right] \right| \tag{2}$$

is negligible. The lemma follows by combining Eqs. (1) and (2) via the triangle inequality. ■

From NIWI to NIZK. Feige *et al.* [FLS99] showed how to transform a NIWI into a NIZK. In fact it suffices that the NIWI be in the CRS model. As we did not define such NIWIs, we state a weaker result, which relies on WI zaps (i.e., 2-message NIWI where the WI property is for *malicious verifiers*) and suffices for our purposes.

Lemma 3.7 ([FLS99])**.** *Assume that one-way functions exist and that every $R \in \mathcal{NP}$ has a WI zap. Then, every $R \in \mathcal{NP}$ has a* NIZK *argument system.*

Feige *et al.* [FLS99] established Lemma 3.7 in the non-uniform setting. For completeness, we include a straightforward adaptation of their proof to the uniform setting in Appendix B.

3.2 Non-interactive commitments

Next, we define non-interactive bit-commitments. We focus on statistical binding commitments in the CRS model, and require the hiding property to hold even wrt a maliciously chosen CRS.

Definition 3.8 (non-interactive bit-commitment)**.** *A* non-interactive bit-commitment scheme in the CRS model *is a ppt algorithm* commit *which gets as input a common random string $crs \in \{0,1\}^\lambda$ (where λ denotes a security parameter) and a bit $b \in \{0,1\}$, and outputs a commitment string. We require:*

- **(Computational hiding:)** *For every pair of ppt adversaries (A_1, A_2) there exists a negligible function δ such that*

$$\left| \Pr_{crs\leftarrow A_1(1^\lambda)} [A_2(crs, \mathsf{commit}(crs,0))=1] - \Pr_{crs\leftarrow A_1(1^\lambda)} [A_2(crs, \mathsf{commit}(crs,1))=1] \right|$$

  is less than $\delta(\lambda)$, where the probability is over the coins of A_1, A_2 and commit *(and A_1 always outputs a crs of length λ).*

- **(Statistical binding:)** *With all but negligible probability in λ over $crs \in \{0,1\}^\lambda$, the distributions* $\mathsf{commit}(crs,0)$ *and* $\mathsf{commit}(crs,1)$ *are disjoint.*

We emphasize that the hiding property holds wrt an adversarial choice of *crs*, whereas the binding property only holds whp over the choice of the *crs*. If $S : \mathbb{N} \to \mathbb{N}$ is a function, we define an *S-secure non-interactive commitment scheme* as one in which every time S adversary (A_1, A_2) attempting to violate the hiding property, can only do so with advantage at most negligible in S (note that the definition of binding is unchanged). A scheme that is 2^{λ^η}-secure for some $\eta > 0$, is called *sub-exponentially secure*.

We extend Definition 3.8 (as well as the sub-exponential variant) to an analogous definition in the *plain model*, by requiring that the common random string is fixed to 1^λ in both the hiding and binding conditions (and in particular, binding should hold with probability 1).

Finally we note that non-interactive commitments in the CRS model exist assuming the existence of one-way functions [HIL+99; Nao91]. Non-interactive commitments in the *plain model* exist assuming *injective* one-way functions [Blu82] (which follow from the hardness of factoring or discrete log) or assuming one-way functions *and* a hitting-set generator against co-nondeterministic circuits [OV07].

3.3 Zero-knowledge PCPs

We next define zero-knowledge PCPs [DFK+92; KPT97] (see [Wei22] for a recent survey). We focus on a non-adaptive definition and restrict our attention to *perfect* zero-knowledge.

Definition 3.9 (zkPCP). *Let $q, q_{\max}, r, \ell \colon \mathbb{N} \to \mathbb{N}$, let $\Sigma = (\Sigma_n)_{n\in\mathbb{N}}$ be an ensemble of sets, and let R be an $\mathcal{NP}$-relation. A $(\delta_S, q, q_{\max}, r, \ell, T_P, T_V)$-zero-knowledge probabilistically checkable proof ($(\delta_S, q, q_{\max}, r, \ell, T_P, T_V)$-zkPCP) for R over an alphabet Σ is a triplet of ppt algorithms (P, V, S) so that the prover P is given as input $x \in \{0,1\}^n$ and a witness w and in time T_P outputs a proof string $\pi \in \Sigma^\ell$, and the verifier V is given as input the same string x and also has oracle access to the symbols of π. The verifier uses at most r random bits and non-adaptively queries q symbols of π and in time T_V either accepts or rejects. We require that:*

1. **(Completeness:)** *If $(x, w) \in R$ then V always accepts given x and oracle access to $\pi = P(x, w)$.*
2. **(Soundness:)** *For every $x \notin L(R)$ and proof-string $\pi^* \in \Sigma^\ell$ it holds that V rejects given input x and oracle access to π^* with all but $\delta_S(|x|)$ probability.*
3. **(Zero-Knowledge:)** *For every $(x, w) \in R$ and set $Q \subseteq [\ell]$ of size at most $q_{\max}(|x|)$, it holds that $S(x, Q)$ is identically distributed to $\pi|_Q$, where $\pi \leftarrow P(x, w)$.*

We remark that the original zkPCP construction of [KPT97] (as well as follow-ups such as [IWY16]) only construct *statistical zero-knowledge* PCPs (i.e., the statistical distance between $S(x, Q)$ and $\pi|_Q$ is small). Unfortunately we cannot rely on such PCPs as we are only able to prove the security of our construction using (near-)perfect zero-knowledge PCPs. Thankfully, a recent construction[26] of Hazay *et al.* [HVW22] (building on the celebrated "MPC in the head" technique of [IKO+09]) constructs such perfect zero-knowledge PCPs. We remark that $q_{\max}$ in their construction depends polynomially on the honest-verifier's query complexity q (in contrast to, say, [KPT97], in which the dependence is exponential), but this suffices for our construction.

Theorem 3.10 (a derandomized version of [HVW22, Theorem 4])**.** *Let $T(n) \geq n$ be a time computable function and let R be a relation for an $\mathcal{NTIME}[T]$ language. Then, for any parameter $q_{\max} = q_{\max}(T) = \omega(\log T)$ the relation R has a $(\delta_S, q, q_{\max}, r, \ell, T_V, T_P)$-zkPCP over an alphabet Σ, where $\delta_S = 1/2$, $q = \tilde{O}(\sqrt{q_{\max}})$, $r = O(\log \ell)$, $\ell = O(q_{\max})$, $\log(|\Sigma|) \leq \mathrm{poly}(T, q_{\max})$, the prover runs in time $T_P = \mathrm{poly}(T, q_{\max})$, and the verifier runs in time $T_V = \mathrm{poly}(T, q_{\max})$.*

Note that the number $q_{\max}$ of allowed queries is nearly-maximal – it is a constant fraction of the proof length. Hazay *et al.* [HVW22] do not precisely prove Theorem 3.10 since their randomness complexity is much larger (i.e., it is $\tilde{O}(\sqrt{\ell})$ rather than the $O(\log \ell)$ stated above). In Appendix A we show a simple derandomization of their construction that achieves the parameters stated in Theorem 3.10.

[26] Actually, [HVW22] give several constructions. Here we use a relatively simple construction which utilizes a rather large alphabet and do not need the more involved constructions over smaller alphabets.

3.4 On the honest prover in cs-$\mathcal{NTIME}$

In this section we prove that two standard observations regarding the power of the honest prover in argument systems extend to cs-$\mathcal{NTIME}$. We first show that the honest prover for $L \in$ cs-$\mathcal{NTIME}$ must be able to solve L by itself, at least on average. Then, we observe that if we allow the honest prover to be stronger than the adversaries, then the definition trivializes (i.e., such systems are unreasonably strong, and for trivial reasons).

3.4.1 Simulating cs-$\mathcal{NTIME}$ in $\mathcal{BPTIME}$

We observe that if a problem is solvable in cs-$\mathcal{NTIME}$ with a T_P-time prover, then it is also contained in $\mathcal{BPTIME}[T_P]$ on average. Specifically, for a distribution ensemble $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ over inputs, we say that $L \in \mathsf{heur}_{\mathbf{x}}\mathcal{BPTIME}[T]$ if there is a probabilistic T-time algorithm A such that for every sufficiently large $n \in \mathbb{N}$, with all but negligible probability over $x \sim \mathbf{x}_n$ it holds that $\Pr[A(x) = L(x)] \geq 2/3$. Then:

Proposition 3.11 (simulating cs-$\mathcal{NTIME}$ in $\mathcal{BPTIME}$)**.** *For any time bounds $T_V < T_P$, and every* poly(T_P)*-time samplable distribution* $\mathbf{x}$*, we have that*

$$\text{cs-}\mathcal{NTIME}[T_V, T_P] \subseteq \mathsf{heur}_{\mathbf{x}}\text{-}\mathcal{BPTIME}[O(T_P)] \,,$$

Proof. Let $L \in$ cs-$\mathcal{NTIME}[T_V, T_P]$. On input x, a probabilistic algorithm A_L for L simulates the honest prover and the verifier, and accepts iff the verifier accepts. Indeed, $x \in L$ will always be accepted. Assume towards a contradiction that there is a poly(T_P)-time distribution $\mathbf{x} = \{\mathbf{x}_n\}$ such that with noticeable probability over $x \sim \mathbf{x}_n$ it holds that $x \notin L$ and A_L accepts x. Then, there is an adversary that breaks the soundness of the cs-$\mathcal{NTIME}$ system, by choosing $x \sim \mathbf{x}_n$ and simulating the honest prover. ■

A corollary of Proposition 3.11 is that, loosely speaking, if a class $\mathcal{C}$ supports worst-case to average-case reduction, and $\mathcal{C} \not\subseteq \mathcal{BPTIME}[T]$, then in any cs-$\mathcal{NTIME}$ system for $\mathcal{C}$, the honest prover will need to run in time at least $\approx T$ (where the only loss is due to the worst-case to average-case reduction). For example:

Corollary 3.12. *Assume that $\mathcal{DSPACE}[O(n)] \not\subseteq \mathcal{BPTIME}[2^{\epsilon\cdot n}]$, for some $\epsilon > 0$. Then, $\mathcal{DSPACE}[O(n)] \not\subseteq$ cs-$\mathcal{NTIME}[2^{o(n)}, 2^{o(n)}]$.*

Proof. Using our assumption and a worst-case to average-case reduction (see [CRT22, Theorem 1.2], following [TV07]), we deduce that there is $L' \in \mathcal{DSPACE}[O(n)]$ such that any probabilistic algorithm running in time $2^{\delta\cdot n}$ (for some $\delta > 0$) computes L' correctly on no more than a 0.51-fraction of the inputs. By Proposition 3.11, L' cannot have a cs-$\mathcal{NTIME}[T_V, 2^{\delta'\cdot n}]$ system, for any $\delta' < \delta$ and $T_V < 2^{\delta'\cdot n}$. ■

Note that Corollary 3.12 explains why the running time of the honest prover in Theorem 1.3 is exponential.

3.4.2 Argument systems in which the honest prover is stronger than adversaries are unreasonably strong

We prove the well-known fact that, under reasonable hardness hypotheses, every language $L \subseteq \{0,1\}^*$ (e.g., the halting problem) can be decided in argument systems in which the honest prover has more computational resources than the adversary. The point is to demonstrate the triviality of this definition, and explain why Definition 1.1 allows the adversary to have strictly more computational resources than the honest prover.

Definition 3.13 (extending Definition 1.1 to allow for excessively strong honest provers)**.** *We say that $L \in$ cs-$\mathcal{NTIME}[T_V, T_P, T_A]$ if there exists a verifier V and a prover P that meet the same conditions as in Definition 1.1, except that the soundness condition now holds only against adversaries $\widetilde{P}$ running in time T_A (instead of* $\mathrm{poly}(T_P)$*).*

Proposition 3.14 (deterministic argument systems with an excessively strong honest prover are trivial)**.** *Assume that there is a unary function f mapping 1^n to n bits such that:*

1. *f is computable in time $T(n)$.*
2. *For every probabilistic algorithm A running in time $T_0(n) < T(n)$, we have $\Pr[A(1^n) = f(1^n)] \leq T_0(n)^{-\omega(1)}$.*
3. *There is a linear-time algorithm deciding the language $L_f = \{(1^n, f(1^n))\}_{n \in \mathbb{N}}$.*

Then, every $L \subseteq \{0,1\}^*$ satisfies $L \in$ cs-$\mathcal{NTIME}[O(n), T, T_0]$.

Proof. On any input $x \in \{0,1\}^n$ (regardless of membership in L), the honest prover sends $f(1^n)$ to the verifier, and the verifier accepts iff $(1^n, f(1^n)) \in L_f$. Note that the completeness condition holds, and that no T_0-time algorithm can find y such that $(1^n, y) \in L_f$, except with negligible probability. ■

The triviality in Proposition 3.14 is not just due to the fact that any language can be decided in the model (i.e., the model is unreasonably strong). The proof system itself is also trivial: It has nothing to do with the language, and the algorithms do not even examine their input. (Thus, it satisfies a stronger soundness condition that holds for every fixed input, as in Remark 1.4.)

The specific assumption in Proposition 3.14 is intended to illustrate the main point, which is robust to various forms of hardness assumptions. For example, the same type of argument shows that, under cryptographic assumptions, every $L \subseteq \{0,1\}^*$ has an argument system in the CRS model with a prover that has more resources than the adversary. The verifier can use the CRS to choose a cryptographic puzzle that requires time T to solve (see, e.g., [DN93; RSW96]), and ask the prover to solve it; the honest prover will succeed in doing so, but time-T_0 adversaries will fail. Again, in such a proof systems the parties do not even examine the input x.

4 Fiat-Shamir for GKR from complexity-theoretic assumptions

In this section we prove results that are obtained by using a hard function f (i.e., a function f hard for small space over all efficiently samplable distributions) to derandomize the proof system of

Goldwasser, Kalai, and Rothblum [GKR15] (equivalently, we instantiate the Fiat-Shamir heuristic for this proof system with f), and show implications of these results.

In Section 4.1 we show how to use a hard function f as above to derandomize the proof system of [GKR15], and deduce Theorems 1.3 and 1.7 as corollaries. In Section 4.2 we show that if a suitable hard function f exists then $\mathcal{NP} \neq \mathcal{PSPACE}$, if true, is inherently non-constructive; and in Section 4.3 we show that if a suitable hard function f exists then $\mathcal{NP} \neq \mathcal{PSPACE}$, if true, is not provable in APC_1.

4.1 Simulating $\mathcal{PSPACE}$ in cs-$\mathcal{NP}$ and $\mathcal{NC}$ in cs-$\mathcal{NTIME}[n^{1+\epsilon}]$

We first prove a general and parameterized result. Specifically, the following assumption is parameterized by time bounds T_P (for the honest prover) and T_V (for the verifier), by the output length $m(n)$ of the hard function, and by a space bound $S(n)$ for the hardness assumption.

Assumption 4.1. *There exists $f\colon \{0,1\}^* \to \{0,1\}^*$ mapping n bits to $m(n)$ bits and a deterministic T_V-time machine M such that:*

- **Upper bound.** *There is an algorithm W running in time T_P such that for every x it holds that $M(x, W(x)) = f(x)$, where $|W(x)| \leq T_V(|x|)$.*
- **Lower bound 1.** *For every algorithm $\tilde{W}$ running in time $\mathrm{poly}(T_P)$ (the polynomial poly may be arbitrarily large) and every sufficiently large $n \in \mathbb{N}$ it holds that*

$$\Pr_{(x,w)\leftarrow\tilde{W}(1^n)}[M(x,w) \notin \{f(x), \perp\}] \leq \mathsf{negl}(T_P) .$$

- **Lower bound 2.** *For every probabilistic algorithm Samp running in time $\mathrm{poly}(T_P)$ (again, the polynomial may be arbitrarily large) and every space-S (deterministic) algorithm A and every sufficiently large $n \in \mathbb{N}$ it holds that*

$$\Pr_{x\leftarrow\mathsf{Samp}(1^n)}[A(x) = f(x)] \leq \mathsf{negl}(T_P) .$$

Indeed, a natural special case of Assumption 4.1 is a function $f \in \mathcal{FP}$ (i.e., $f(x)$ is efficiently computable without a witness $W(x)$, in which case there is no need to lower bound the adversarial $\tilde{W}$'s), and this special case also makes for an appealing assumption. We consider the more relaxed version in Assumption 4.1 since it will be *equivalent* to $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$.

We now show how a function as in Assumption 4.1 can be used to derandomize the proof system of [GKR15] (equivalently, to instantiate the Fiat-Shamir heuristic for this proof system). We state a general and parameterized result, and later deduce special cases as corollaries. When parsing the parameters below, note that $\bar{D}$ is the (maximal) length of the partial transcripts on which we will apply the hard function f; in fact, to relax the hardness assumption we will use padding, so $\bar{D}$ is the length of the padded partial transcript, where the constant $c > 1$ controls the amount of padding. Also, think of the constant $\epsilon > 0$ as arbitrarily small.

Theorem 4.2 (Fiat-Shamir for GKR in the plain model from complexity-theoretic assumptions)**.** *Let $L \subseteq \{0,1\}^*$ be decidable by logspace-uniform circuits of size T and depth d, let $\epsilon \in (0,1)$ and $c \geq 1$*

be constants, and let $\bar{D} = ((n \cdot d)^{1+\epsilon} \cdot \log(T)^3)^c$. *Suppose that Assumption 4.1 holds with* T_P, T_V, S, m *such that*

$$\begin{aligned} \text{poly}(T) &\leq T_P(\bar{D}) \\ \log(n) &\leq m(n) \leq n^{\epsilon/c'} \\ S(\bar{D}) &= \Theta(\log(T) \cdot (m(\bar{D}) + d)) \ , \end{aligned}$$

where $c' > 1$ *is a large enough universal constant. Then,*

$$L \in \text{cs-}\mathcal{NTIME}\left[n^{1+\epsilon} + d \cdot \text{polylog}(T) \cdot n^{\epsilon} \cdot T_V(\bar{D}),\ \ \text{poly}(T_P(\bar{D}))\right] \ .$$

Proof. Let $\{C_n\}_{n\in\mathbb{N}}$ be a logspace-uniform circuit family of size T and depth d deciding L. Let $\{C'_n\}_{n\in\mathbb{N}}$ be the modification of $\{C_n\}$ by Goldreich [Gol18], which is of size $T' = 2^{O(\log T)}$ and depth $d' = O(d \cdot \log(T))$. We will later use the fact that the family $\{C'_n\}$ computes the same function as $\{C_n\}$ but is more uniform than $\{C_n\}$; specifically, the circuit-structure function of $\{C'_n\}$ (i.e., the Boolean function that gets input (w, u, v) and outputs 1 if and only if u and v are the two children of w) is computable by Boolean formulas of size $O(\log T)$ that can be printed in time $\text{polylog}(T)$ and space $O(\log T)$ (see [KLV23, Fact 5.4] and [CRT22, Proof of Claim 4.2.2]).

The GKR protocol for $\{C'_n\}$**.** On input $x \in \{0,1\}^n$, let $\alpha_i \colon [T'] \to \{0,1\}$ be the function representing the gate-values of $C'_n(x)$ at layer i, where $i = 1$ is the output layer and $i = d' + 1$ represents the input layer.[27]

For $\ell = \log(T')$ and $i \in [d']$, let $\Phi_i \colon \{0,1\}^{3\ell} \to \{0,1\}$ be the circuit-structure function of the i^{th} layer of $\{C'_n\}$ (i.e., $\Phi_i(\vec{w}, \vec{u}, \vec{v}) = 1$ if and only if the gate indexed by $\vec{w}$ in layer i is fed by gates $\vec{u}, \vec{v}$ in layer $i+1$). Kalai, Lombardi, and Vaikuntanathan [KLV23, Lemma 5.3] defined the "variable-extended formulation" $\Phi'_i \colon \{0,1\}^{3\ell + c\cdot\log(T)}$ of Φ_i (where $c > 1$ is constant), which has the following properties:

Fact 4.2.1. *The set* $\{\Phi'_i\}_{i\in[d']}$ *has the following properties.*

1. *For every* $(\vec{w}, \vec{u}, \vec{v}) \in \{0,1\}^{3\ell}$ *such that* $\Phi_i(\vec{w}, \vec{u}, \vec{v}) = 1$ *there is a unique* $\vec{z} \in \{0,1\}^{c\cdot\log(T)}$ *such that* $\Phi'_i(\vec{w}, \vec{u}, \vec{v}, \vec{z}) = 1$.
2. *There is an algorithm that gets input* $n \in \mathbb{N}$ *and* $i \in [d']$, *runs in time* $\text{polylog}(T)$ *and space* $O(\log T)$, *and prints an arithmetic circuit over* $\mathbb{F}_2$ *of size* $O(\log T)$ *and individual degree two computing* Φ'_i.

Proof. The only property that is not explicitly stated in [KLV23] is that the algorithm printing an arithmetic circuit for Φ'_i runs in space $O(\log T)$. This follows immediately from the construction of the circuit in [KLV23, Lemma 5.3], while relying on the fact (mentioned above) that the circuit for Φ_i from Goldreich's construction of $\{C'_n\}$ is printable in time $\text{polylog}(T)$ and space $O(\log T)$. □

[27] That is, for the input layer we have $\alpha_{d'+1}(j) = x_j$ for $j \in [n]$ and $\alpha_{d'+1}(j) = 0$ for $j \in [T'] \setminus [n]$; and for the output layer we have $\alpha_1(1) = C'_n(x) = L(x)$ and $\alpha_1(j) = 0$ for $j \in [T'] \setminus \{1\}$.

Relying on Fact 4.2.1, for every $\vec{w} \in \{0,1\}^\ell$ we have

$$\begin{aligned}\alpha_i(\vec{w}) &= \sum_{\vec{u},\vec{v}\in\{0,1\}^\ell} \Phi_i(\vec{w},\vec{u},\vec{v}) \cdot (1-\hat{\alpha}_{i+1}(\vec{u})\cdot\hat{\alpha}_{i+1}(\vec{v})) \\ &= \sum_{\substack{\vec{u},\vec{v}\in\{0,1\}^\ell \\ \vec{z}\in\{0,1\}^{c\cdot\log(T)}}} \Phi'_i(\vec{w},\vec{u},\vec{v},\vec{z}) \cdot (1-\hat{\alpha}_{i+1}(\vec{u})\cdot\hat{\alpha}_{i+1}(\vec{v}))\ .\end{aligned}$$

Let $\mathbb{F}$ be a field of size $2^{m(\bar{D})}$. Let $\hat{\Phi}_i\colon \mathbb{F}^{3\ell+c\cdot\log(T)} \to \mathbb{F}$ be the extension of Φ'_i of individual degree 2 that is computed by the arithmetic formula for Φ'_i. (Recall that the arithmetic formula is over $\mathbb{F}_2 \subseteq \mathbb{F}$.) Let $\hat{\alpha}_{d'+1}\colon \mathbb{F}^\ell \to \mathbb{F}$ be the unique multilinear extension of the function $x\colon \{0,1\}^{\log(n)} \to \{0,1\}$ defined by the input x.[28] For $i \in [d']$, we define $\hat{\alpha}_i\colon \mathbb{F}^\ell \to \mathbb{F}$ by

$$\hat{\alpha}_i(\vec{w}) = \sum_{\substack{\vec{u},\vec{v}\in\{0,1\}^\ell \\ \vec{z}\in\{0,1\}^{c\cdot\log(T)}}} \hat{\Phi}_i(\vec{w},\vec{u},\vec{v},\vec{z}) \cdot (1-\hat{\alpha}_{i+1}(\vec{u})\cdot\hat{\alpha}_{i+1}(\vec{v}))\ , \tag{4.1}$$

and observe that $\hat{\alpha}_i$ is an extension of α_i of individual degree at most two.

The protocol is divided into d' phases, where each phase reduces a claim about two values in the i^{th} layer to a claim about two values in the $(i+1)^{th}$ layer.

The i^{th} phase: For $i = 1,...,d'-1$, we enter the i^{th} phase with two claims of the form "$\hat{\alpha}_i(\vec{w}_\sigma^{(i)}) = v_\sigma^{(i)}$", where $\sigma \in \{1,2\}$. In the first phase $i = 1$, the claims refer to the value of the output, i.e. to $C'_n(x)$; that is, the protocol opens with the prover sending the claim "$\hat{\alpha}_i(0^\ell) = b$", where $b \in \{0,1\} \subseteq \mathbb{F}$ and 0^ℓ represents the index of the output gate. (And to be consistent with subsequent phases, we consider this to be two identical claims.)

Let $P_i\colon \mathbb{F}^{3\ell+c\cdot\log(T)} \to \mathbb{F}$ be defined by

$$P_i(\vec{w},\vec{u},\vec{v},\vec{z}) = \hat{\Phi}_i(\vec{w},\vec{u},\vec{v},\vec{z}) \cdot (1-\hat{\alpha}_{i+1}(\vec{u})\cdot\hat{\alpha}_{i+1}(\vec{v}))\ , \tag{4.2}$$

and note that the claim "$\hat{\alpha}_i(\vec{w}_\sigma^{(i)}) = v_\sigma^{(i)}$" is equivalent to the claim

$$\sum_{(\vec{u},\vec{v},\vec{z})\in\{0,1\}^{2\ell+c\cdot\log(T)}} P_i(\vec{w}_\sigma^{(i)},\vec{u},\vec{v},\vec{z}) = v_\sigma^{(i)}\ . \tag{4.3}$$

In the i^{th} phase, the verifier and prover use two sumcheck protocols to reduce the two claims in Eq. (4.3) (i.e., for $\sigma = 1,2$) to two claims of the form $P_i(\vec{w}_\sigma^{(i)},\vec{a},\vec{b},\vec{c}) = r_\sigma$, for some fixed $(\vec{a},\vec{b},\vec{c}) \in \mathbb{F}^{2\ell+c\cdot\log(T)}$ chosen by the verifier and $r_1, r_2 \in \mathbb{F}$. Crucially, the verifier uses the same random choices for both sumcheck protocols, and thus both protocols yield the same choices of $\vec{a},\vec{b},\vec{c}$.

Thus, at the end of the phase we have the two claims

$$\hat{\Phi}_i(\vec{w}_\sigma^{(i)},\vec{a},\vec{b},\vec{c}) \cdot (1-\hat{\alpha}_{i+1}(\vec{a})\cdot\hat{\alpha}_{i+1}(\vec{b})) = r_\sigma\ , \tag{4.4}$$

[28] In more detail, we consider the function $x'\colon \{0,1\}^\ell \to \{0,1\}$ such that $x'(i) = x_i$ for $i \in [n]$ and $x'(i) = 0$ for $i \in \{n+1,...,T'\}$, and $\hat{\alpha}_{d'+1}$ is the unique multilinear extension of x'.

for $\sigma = 1, 2$. The prover sends values $v_1^{(i+1)}, v_2^{(i+1)} \in \mathbb{F}$ such that, supposedly, $\hat{\alpha}_{i+1}(\vec{a}) = v_1^{(i+1)}$ and $\hat{\alpha}_{i+1}(\vec{b}) = v_2^{(i+1)}$. The verifier checks that Eq. (4.4) holds for $\sigma = 1, 2$, and if so, it proceeds to the $(i+1)^{th}$ phase with $\vec{w}_1^{(i+1)} = \vec{a}$ and $\vec{w}_2^{(i+1)} = \vec{b}$.

The d^{th} (last) phase. This phase is identical to a generic phase, except that after the sumcheck protocol, the verifier remains with the two claims

$$\hat{\Phi}_i(\vec{w}_\sigma^{(d')}, \vec{a}, \vec{b}, \vec{c}) \cdot (1 - \hat{\alpha}_{d'+1}(\vec{a}) \cdot \hat{\alpha}_{d'+1}(\vec{b})) = r_\sigma \ ,$$

and it can check the two claims directly by computing the multilinear extension $\hat{\alpha}_{d'+1}$ of the input x at $\vec{a}$ and $\vec{b}$.

Analysis. The analysis follows [KLV23, Section 5]. At a high level, the only differences between this analysis and the original analysis of Goldwasser, Kalai, and Rothblum [GKR15] (as simplified by Goldreich [Gol18]) are that each univariate sent by the prover is of constant degree (rather than a fixed polynomial degree as in [GKR15]), and that in each phase there are two sumcheck protocols reducing two claims to two claims (replacing the 2-to-1 trick of [GKR15]).

For a phase $i \in [d']$, and a round $j \in [2\ell + c \cdot \log(T)]$ of sumcheck, let $a_1, ..., a_{j-1} \in \mathbb{F}$ be the elements sent in the previous $j-1$ rounds during phase i. For $\sigma = 1, 2$, we denote by $p_{i,j,\sigma}^{\mathsf{correct}} \colon \mathbb{F} \to \mathbb{F}$ the correct sumcheck polynomial; that is,

$$p_{i,j,\sigma}^{\mathsf{correct}}(a_j) = \sum_{a_{j+1},...,a_{2\ell+c\cdot\log(T)} \in \{0,1\}} P_i(\vec{w}_\sigma^{(i)}, a_1, ..., a_{2\ell+c\cdot\log(T)}) \ , \tag{4.5}$$

where $P_i(\vec{w}, \vec{u}, \vec{v}, \vec{z})$ is as in Eq. (4.2).

We now highlight the key facts of the analysis that we will need later:

Fact 4.2.2. *The protocol has d' phases, and each phase consists of $2\ell + c \cdot \log(T)$ rounds, for a total of $R = d' \cdot (2\ell + c \cdot \log(T))$ rounds. In each round, the prover sends two univariate polynomials of degree four, corresponding to the two sumcheck protocols, and the verifier responds with a (single) random field element.*

Proof. The only non-trivial part is that the univariates sent by the prover are of degree at most four. To see this, observe that for $i \in [d']$ the polynomial $\hat{\alpha}_i$ is of individual degree two (since it only feeds its input into $\hat{\Phi}_i$), and that $\hat{\alpha}_{d'+1}$ is multilinear; and note that each sumcheck polynomial feeds its input into $\hat{\Phi}_i$ and into $\hat{\alpha}_{i+1}$ (see Eq. (4.2)). □

Fact 4.2.3. *The total length of the transcript is at most $D \stackrel{\text{def}}{=} \left((d \cdot n)^{1+\epsilon/2} \cdot \log(T)^{2+\epsilon}\right)^c$.*

Proof. In each of the R rounds, the verifier and prover send constantly many field elements. Thus, the transcript length (also accounting for the input x) is at most

$$\begin{aligned} R \cdot O(\log(|\mathbb{F}|)) &\leq O(n \cdot d' \cdot \log(T)) \cdot m \left(((n \cdot d)^{1+\epsilon} \log(T)^3)^c\right) \\ &\leq O(n \cdot d \cdot \log(T)^2) \cdot ((n \cdot d)^{1+\epsilon} \log(T)^3)^{\epsilon \cdot c/c'} \\ &< O(n \cdot d \cdot \log(T)^2 \cdot (n \cdot d)^{(1+\epsilon)\cdot(\epsilon/c')} \log(T)^{3\epsilon/c'})^c \\ &\leq ((d \cdot n)^{1+\epsilon/2} \cdot \log(T)^{2+\epsilon})^c \ , \end{aligned}$$

where we relied on the fact that $m(\bar{D}) \leq \bar{D}^{\epsilon/c'}$, on the fact that $\epsilon < 1$, and on $c' > 1$ being sufficiently large. □

Fact 4.2.4. *For $i \in [d'-1]$, if we enter a phase i with an incorrect claim, and the verifier does not reject at the end of the phase, then at least one of two things happened:*

1. *We enter the next phase $i+1$ with an incorrect claim.*
2. *At some round j of sumcheck during the phase, the verifier chose a field element that for some $\sigma \in \{1,2\}$ is one of the roots of the univariate polynomial $p_{i,j,\sigma} - p^{\text{correct}}_{i,j,\sigma}$ of degree at most four, where $p_{i,j,\sigma} \colon \mathbb{F} \to \mathbb{F}$ is the polynomial sent by the prover.*

For $i = d'$, if we enter phase d' with an incorrect claim and the verifier does not reject, then Item (2) happened.

Proof. This follows from a standard analysis of the sumcheck protocol. □

Fact 4.2.5. *For each fixed (i,j,σ), the probability that the verifier chooses one of the roots of $p_{i,j,\sigma} - p^{\text{correct}}_{i,j,\sigma}$ is at most $\frac{4}{(d\cdot n)^{1+\epsilon}\cdot\log(T)^3}$.*

Proof. This follows since $p^{\text{correct}}_{i,j,\sigma}$ and $p_{i,j,\sigma}$ are of degree four, and since

$$4/|\mathbb{F}| = 4 \cdot 2^{-m(\bar{D})} \leq \frac{4}{((d\cdot n)^{1+\epsilon}\cdot\log(T)^3)^c} \leq \frac{4}{(d\cdot n)^{1+\epsilon}\cdot\log(T)^3}\,,$$

where we relied on the facts $m(\bar{D}) \geq \log(\bar{D})$ and $c \geq 1$. □

Note that the error probability in Fact 4.2.5 is small enough for a union-bound over the $2R = O(d\cdot\log(T)^2)$ choices of field elements in the protocol (each choice corresponding to some (i,j,σ)).

A derandomized version. Let us define a derandomized verifier V for the protocol above. Recall that $\bar{D} = ((n\cdot d)^{1+\epsilon}\cdot\log(T)^3)^c$ and observe that $\bar{D} > D$ and that on inputs of length $\bar{D}$, the output length of f is $m(\bar{D}) = \log(|\mathbb{F}|)$.

In each phase $i = 1,...,d'$, and for each round $j = 1,...,2\ell + c\cdot\log(T)$ of sumcheck, instead of choosing a field element at random, the verifier V acts as follows. Let $\pi_{i,j}$ be the description of the polynomials and field elements sent up to that point, and pad $\pi_{i,j}$ with zeroes to obtain $\bar{\pi}_{i,j}$ of length $\bar{D}$. The verifier expects to receive $w_{i,j}$ from the prover, and computes $r_{i,j} = M(\bar{\pi}_{i,j}, w_{i,j})$, where M is the machine that computes f. If $r_{i,j} = \perp$ or $|r_{i,j}| \neq \log(|\mathbb{F}|)$ the verifier rejects, and otherwise it treats $r_{i,j}$ as a field element, and sends it to the prover. We stress that the string $\pi_{i,j}$ does not include the witnesses sent from the prover to the verifier in previous rounds; the string $\pi_{i,j}$ is the non-contiguous substring of the partial transcript that includes only the univariate polynomials sent by the prover and the field elements sent by the verifier (i.e., $\pi_{i,j}$ excludes the witnesses $w_{i,j}$ sent by the prover for the computation of f).

Note that the prover can compute the transcript for the entire interaction in advance, and send it as a single message to the verifier. This yields a deterministic one-message protocol (i.e., an $\mathcal{NP}$-type machine), with a verifier running in time

$$\begin{aligned}\text{polylog}(|\mathbb{F}|)\cdot\left(\tilde{O}(n) + R\cdot T_V(\bar{D})\right) &\leq \text{poly}(m(\bar{D}))\cdot\left(\tilde{O}(n) + d'\cdot\log(T)\cdot T_V(\bar{D})\right)\\ &< (n + d\cdot\text{polylog}(T)\cdot T_V(\bar{D}))\cdot(n\cdot d\cdot\log(T))^{\epsilon}\,,\end{aligned}$$

where the polylog($|\mathbb{F}|$) term comes from performing field operations, and the $\tilde{O}(n)$ factor is the time required to compute the multilinear extension of the input x.

Completeness. The honest prover has two tasks in each round. The first task is to compute a representation of the correct sumcheck polynomials $p^{\text{correct}}_{i,j,\sigma}$ as in Eq. (4.5). Since $p^{\text{correct}}_{i,j,\sigma}$ is of constant degree, it suffices to interpolate it using constantly many points, and this task reduces to computing P_i at poly(T) points. This can be done in time poly(T). The second task of the honest prover in each round is to compute a witness $w_{i,j}$ for $M(\bar{\pi}_{i,j}, \cdot)$, which can be done in time $T_P(|\bar{\pi}_{i,j}|)$ (using the machine W from Assumption 4.1). This yields an honest prover that runs in total time

$$R \cdot \underbrace{\text{poly}(T(n))}_{\text{GKR prover}} \cdot \underbrace{T_P(\bar{D})}_{\text{the machine } W} \leq \text{poly}(T_P(\bar{D})) \ .$$

When interacting with this honest prover, the verifier always accepts.

Soundness. Assume towards a contradiction that there is a probabilistic algorithm P running in time poly$(T_P(\bar{D}))$ and an infinite set $N \subseteq \mathbb{N}$ of input lengths such that for every $n \in N$, with noticeable probability $T_P(\bar{D})^{-O(1)}$ over $(x, \pi) \leftarrow P(1^n)$ it holds that $V(x, \pi) \notin \{\bot, C_n(x)\}$.

Fix any pair (x, π) such that $V(x, \pi) \notin \{\bot, C_n(x)\}$. Recall that π consists of a sequence of $2R$ univariate polynomials $\{p_{i,j,\sigma}\}$ of degree four, and a sequence of R witnesses $\{w_{i,j}\}$ for the computations of M. Since we enter the first phase with an incorrect claim, and the verifier does not reject after the last phase, by Fact 4.2.4 there exists a minimal $i \in [d']$ such that we enter the i^{th} phase with an incorrect claim but enter the $(i+1)^{th}$ phase with a correct claim. For this i, by Fact 4.2.4 again, there is some j and σ such that $r_{i,j}$ is a root of $p_{i,j,\sigma} - p^{\text{correct}}_{i,j,\sigma}$.

Fact 4.2.6. *Fix any (x, π) such that $V(x, \pi) \notin \{\bot, C_n(x)\}$, and let (i, j, σ) be defined as above. Then, one of the following two cases holds:*

1. *$r_{i,j} = f(\bar{\pi}_{i,j})$. In this case, $f(\bar{\pi}_{i,j})$ is a root of $p_{i,j,\sigma} - p^{\text{correct}}_{i,j,\sigma}$. Since this polynomial has degree at most four, we know that one of these four roots must equal $f(\bar{\pi}_i)$.*
2. *$r_{i,j} \notin \{\bot, f(\bar{\pi}_{i,j})\}$. In this case $w_{i,j}$ is such that $M(\bar{\pi}_{i,j}, w_{i,j}) \notin \{\bot, f(\bar{\pi}_{i,j})\}$.*

Since N is infinite, for some $\sigma' \in \{1, 2\}$ there exists an infinite set $N' \subseteq N$ such that for every $n \in N'$, with noticeable probability $T_P(\bar{D})^{-O(1)}$ over $(x, \pi) \leftarrow P(1^n)$, the case in Item σ' of Fact 4.2.6 holds.

Case 1: $\sigma' = 2$. Then, there is an efficient algorithm that, when given 1^n for $n \in N'$, with noticeable probability finds inputs $\bar{\pi}_{i,j}$ and witnesses $w_{i,j}$ such that $M(\bar{\pi}_{i,j}, w_{i,j}) \notin \{\bot, f(\bar{\pi}_{i,j})\}$. Specifically, the algorithm simulates P, chooses i, j uniformly at random, pads $\pi_{i,j}$ to $\bar{\pi}_{i,j}$, and prints $(\bar{\pi}_{i,j}, w_{i,j})$. This algorithm runs in time poly$(T_P(\bar{D}))$, contradicting Assumption 4.1.

Case 2: $\sigma' = 1$. We will define eight linear-space machines, and claim that for one of these machines the following holds: There is an efficient algorithm that for infinitely many values of n, when the algorithm gets input 1^n, with noticeable probability it finds input $\bar{\pi}_{i,j}$ such that the linear-space machine successfully computes $f(\bar{\pi}_{i,j})$. Specifically, the algorithm simulates P, chooses (i, j) uniformly at random, and outputs $\bar{\pi}_{i,j}$; this algorithm also runs in time poly$(T_P(\bar{D}))$.

Turning to the linear-space machines, for each $(k,\sigma) \in [4] \times [2]$, let $M_{k,\sigma}$ be the following procedure. Given input $\bar{\pi}_{i,j}$, it finds the correct sumcheck polynomial $p^{\mathsf{correct}}_{i,j,\sigma}$, enumerates over $\mathbb{F}$ to find the roots of $p^{\mathsf{correct}}_{i,j,\sigma} - p_{i,j,\sigma}$, and outputs the k^{th} root (if it exists). Note that whenever (x,π) is such that $V(x,\pi) \notin \{\perp, C_n(x)\}$, and i,j,k,σ are correct,[29] the procedure $M_{k,\sigma}$ correctly maps $\bar{\pi}_{i,j}$ to $f(\bar{\pi}_{i,j})$. We implement these procedure in low space:

Fact 4.2.7. *Each $M_{k,\sigma}$ can be implemented by a machine running in space $O(\log T) \cdot (m(\bar{D}) + d)$ on inputs of length $\bar{D}$.*

Proof. We first argue that, given the transcript, we can evaluate $p^{\mathsf{correct}}_{i,j,\sigma}$ at any given point using space

$$O((d + \log(|\mathbb{F}|)) \cdot \log(T)) \,.$$

To see this, note that evaluating $p^{\mathsf{correct}}_{i,j,\sigma}$ reduces to computing $\hat{\Phi}_i$ and $\hat{\alpha}_{i+1}$ at $\mathrm{poly}(T)$ points (see Eq. (4.5) and Eq. (4.2)). Then:

- Computing $\hat{\Phi}_i$: Since the arithmetic circuit for Φ'_i can be constructed in space $O(\log T)$ and is of size $O(\log T)$ (see Fact 4.2.1), it can be evaluated over $\mathbb{F}_2$ in space $O(\log T)$ (using the standard DFS-style simulation). By evaluating the same circuit over $\mathbb{F}$, we can compute $\hat{\Phi}_i$ in space $O(\log(T) \cdot \log(|\mathbb{F}|))$.
- Computing $\hat{\alpha}_{i+1}$: When $i = d'$, we can compute $\hat{\alpha}_{d'+1}$ (i.e., the multilinear extension of x) in space $O(\log T)$. For $i < d'$, computing $\hat{\alpha}_{i+1}$ reduces in space $O(\log T)$ to computing $\hat{\Phi}_i$ and to computing $\hat{\alpha}_{i+2}$ on points in $\{0,1\}^\ell$ (see Eq. (4.1)). Since $\hat{\alpha}_{i+2}$ is an extension of α_{i+2}, computing it on points in $\{0,1\}^\ell$ reduces to computing a gate in $C'_n(x)$; using the standard DFS-style simulation of C'_n, this can be done in space $O(d' + \log(T))$.

Thus, computing $\hat{\Phi}_i$ and $\hat{\alpha}_{i+1}$ at any given point can be done in space $O((d + \log(|\mathbb{F}|)) \cdot \log(T))$, and the space complexity of computing $p^{\mathsf{correct}}_{i,j,\sigma}$ follows.

Enumerating over $\mathbb{F}$ and evaluating $p^{\mathsf{correct}}_{i,j,\sigma} - p_{i,j,\sigma}$ can thus be done in space

$$O(\log(T) \cdot (\log(|\mathbb{F}|) + d) + \log(\bar{D})) = O(\log T) \cdot (m(\bar{D}) + d) \,,$$

as we wanted. □

Finally, since N' is infinite, for some $(k,\sigma) \in [4] \times [2]$ there is an infinite set $N'' \subseteq N'$ such that for every $n \in N''$, with noticeable probability over $(x,\pi) \leftarrow P(1^n)$ it holds that $f(\bar{\pi}_{i,j})$ is the k^{th} root of $p_{i,j,\sigma} - p^{\mathsf{correct}}_{i,j,\sigma}$. Thus, on infinitely many input lengths $n \in N''$, the algorithm we defined above finds, with noticeable probability, inputs on which the machine for $M_{k,\sigma}$ correctly computes f. ■

Let us now state Theorem 1.3 formally and prove it. In fact, we state a more general technical result, which asserts an additional equivalence of $\mathcal{PSPACE} \subseteq \mathtt{cs}\text{-}\mathcal{NP}$ to hardness in $\mathtt{cs}\text{-}\mathcal{NP}$ for small space on all but finitely many inputs. One direction of the proof (i.e., assuming a hard function and deducing $\mathcal{PSPACE} \subseteq \mathtt{cs}\text{-}\mathcal{NP}$) follows as a special case of Theorem 4.2, and the proof below focuses on showing that $\mathcal{PSPACE} \subseteq \mathtt{cs}\text{-}\mathcal{NP}$ implies a hard function.

[29] That is, i and j are the ones defined in the paragraph before Fact 4.2.6, and $r_{i,j}$ is the k^{th} root of $p_{i,j,\sigma} - p^{\mathsf{correct}}_{i,j,\sigma}$.

Theorem 4.3 ($\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$; Theorem 1.3, restated)**.** *There is a universal constant $c' > 1$ such that the following statements are equivalent:*

1. *There are $C > 1$ and $\epsilon \in (0, 1/c')$ and $\delta > 0$ such that Assumption 4.1 holds with $m(n) = n^\epsilon$ and $T_P(n) = 2^{n^\delta}$ and $T_V = n^C$ and $S(n) = n^{c' \cdot \epsilon}$.*
2. *For every $L \in \mathcal{PSPACE}$ there are constants $C > c > 1$ such that $L \in$ cs-$\mathcal{NTIME}[n^C, 2^{n^c}]$.*
3. *For any $\epsilon > 0$ there are $\delta > 0$ and $C > 1$ such that there is a function $f \in$ cs-$\mathcal{NTIME}[n^C, 2^{n^\delta}]$ mapping n bits to n^ϵ bits that is hard for space $n^{c' \cdot \epsilon}$ on all but finitely many inputs.*

As explained after the statement of Theorem 1.3, the most appealing setting of parameters for the hardness assumption in Item (1) of Theorem 4.3 is $\delta < \epsilon$, since such hardness is attained by a random function. To get an equivalence, in Theorem 4.3 we also allow for other values of $\delta > 0$, and in particular we also allow taking $\delta > 1$. Indeed, assuming $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$, we will show a function in cs-$\mathcal{NTIME}[\text{poly}(n), 2^{n^\delta}]$ for $\delta > 1$ that is hard for small space on all but finitely many inputs; this function will be obtained by diagonalization, and it is thus quite different from a random function (i.e., both in its parameters and in its properties).

In other words, we show that a function in cs-$\mathcal{NP}$ with hardness that is attained by a random function (i.e., Item (1) with $\delta < \epsilon$) implies that $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$, and we show that the latter containment implies a function in cs-$\mathcal{NP}$ with diagonalization-type hardness (i.e., Item (3), which implies Item (1) with $\delta > 1 > \epsilon$).

Proof of Theorem 4.3. To see that Item (1) implies Item (2), recall that every language in $\mathcal{PSPACE}$ can be decided by a logspace-uniform circuit of size $T(n) = 2^{n^a}$ and depth $d(n) = n^b$, for some constants $a = a_L > 1$ and $b = b_L > 1$. We use Theorem 4.2 with $c = O((a+b)/\epsilon + a/\delta)$ and with the parameter $(c' \cdot \epsilon) \in (0, 1)$, and observe that the required hardness is for space

$$S(\bar{D}) = \Theta(\log(T) \cdot (m(\bar{D}) + d)) \leq O(n^a \cdot \bar{D}^\epsilon + n^{a+b}) \leq \bar{D}^{c' \cdot \epsilon} \ ,$$

where $\bar{D} = ((n \cdot d)^{1+c' \cdot \epsilon} \cdot \log(T)^3)^c$ is as specified in the statement of Theorem 4.2 (recall that $\bar{D}$ is the padded transcript length on which the hard function is applied). The hypothesis in Theorem 4.2 requires that $\text{poly}(T) \leq T_P(\bar{D})$, which with our choices of $T, T_P, \bar{D}$ is equivalent to $2^{\Theta(n^a)} \leq 2^{\bar{D}^\delta}$; this is satisfied because $\bar{D}^\delta > n^{2a}$.

Item (3) trivially implies Item (1), so we just need to show that Item (2) implies Item (3). Let $\epsilon > 0$, and consider the language L that consists of pairs $(x, i) \in [n] \times [n^\epsilon]$ such that the following holds. Let M_i be the i^{th} Turing machine according to some predetermined enumeration, modified so that it runs in space $S = n^{\epsilon \cdot c'}$. Then, $(x, i) \in L$ if and only if $M_i(x)$ halts after $2^{O(S)}$ steps, prints at least i bits, and its i^{th} output bit is 1. Note that $L \in \mathcal{PSPACE}$, and thus $L \in$ cs-$\mathcal{NTIME}[n^{C_0}, 2^{n^\delta}]$ for some constants $C_0 > \delta > 1$.

Consider the function f that maps n bits to $m(n) = n^\epsilon$ bits such that $f(x)_i = 1$ if and only if $(x, i) \notin L$. Let W be the algorithm that, given x, simulates the 2^{n^δ}-time algorithm for L on inputs $(x, 1), ..., (x, m(n))$ and produces a concatenated witness $\bar{w} = w_1, ..., w_{m(n)} \in \{0,1\}^{n^{C_0} \cdot m(n)}$; note that W runs in time $O(2^{n^\delta} \cdot m(n)) < 2^{O(n^\delta)}$. Let M be the algorithm that gets input x and a concatenated witness $\bar{w}$, simulates the n^{C_0}-time machine for L on inputs $\{(x_i, w_i)\}_{i \in [m(n)]}$, and prints the concatenation of the outputs of the latter machine (or $\perp$, if the latter machine outputs $\perp$ on some (x_i, w_i)).

Observe that the algorithm W runs in time at most $T_P = 2^{O(n^\delta)}$ and the machine M runs in time $T_V = O(n^{C_0} \cdot m(n))$. By the properties of the cs-$\mathcal{NTIME}$ system for L, the function f satisfies the completeness requirement of cs-$\mathcal{NTIME}[T_P, T_V]$. Next, we argue soundness:

Fact 4.3.1. *For every algorithm $\tilde{W}$ running in time $\mathrm{poly}(T_P)$ and every sufficiently large n, the probability over $(x,w) \leftarrow \tilde{W}(1^n)$ that $M(x,w) \notin \{f(x), \perp\}$ is negligible in T_P.*

Proof. Assume towards a contradiction that there is $\tilde{W}$ running in time $\mathrm{poly}(T_P)$ such that $\Pr_{(x,w)\leftarrow\tilde{W}(1^n)}[M(x,w) \notin \{f(x),\perp\}] \geq 1/\mathrm{poly}(T_P)$. Note that for every (x,w) such that $M(x,w) \notin \{f(x),\perp\}$ there exists $i = i_{x,w}$ such that the machine M_L for L is incorrect when given input (x,i) and the i^{th} part of the witness w (recall that M_L parses its witness as $m(n)$ concatenated witnesses $w^{(1)}, ..., w^{(m(n))}$).

Consider the algorithm F that simulates $\tilde{W}$ to get (x,w), chooses $i \in [m(n)]$ uniformly at random, and outputs the input (x,i) for L and the i^{th} part of w. With probability at least $1/\mathrm{poly}(T_P) \cdot m(n)^{-1} = 1/\mathrm{poly}(T_P)$ this algorithm yields an input (x,i) for L' and a witness $w^{(i)}$ such that $M_L((x,i), w^{(i)}) \notin \{L(x,i), \perp\}$. □

To conclude the proof we argue that for every space-$(n^{c'\cdot\epsilon})$ machine A and sufficiently large input length n, for every input $x \in \{0,1\}^n$ it holds that $A(x) \neq f(x)$. Let i be the index of A in the enumeration of machines, consider a sufficiently large input length n such that $m(n) \geq i$, and fix any $x \in \{0,1\}^n$. Then, either $A(x)$ does not print $m(n)$ bits, or $A(x)_i \neq f(x)_i$ by the definitions of f and of L. ■

Lastly, we present the simulation of logspace-uniform $\mathcal{NC}$ circuits by cs-$\mathcal{NTIME}$ verifiers running in near-linear time (i.e., Theorem 1.7). Our assumption is a function computable in near-linear time that is hard for polylogarithmic space over all polynomial-time samplable distributions. As in Theorem 4.3, we can also relax the upper bound, and only require that the function is verifiable in near-linear time with computational soundness; that is:

Corollary 4.4 ($\mathcal{NC} \subseteq$ cs-$\mathcal{NTIME}[n^{1+\epsilon}]$ from hardness for small space algorithms)**.** *Let $d(n) = \mathrm{polylog}(n)$ and let $\epsilon \in (0,1)$. Suppose that Assumption 4.1 holds with $T_P = \mathrm{poly}(n)$ and $T_V = n^{1+\epsilon}$ and $m(n) = \mathrm{polylog}(n)$ and $S(n) = \Theta(\log(n) \cdot (d(n) + m(n)))$. Then, for every problem $L \subseteq \{0,1\}^*$ decidable by logspace-uniform $\mathcal{NC}$ circuits of size $\mathrm{poly}(n)$ and depth $d(n)$ it holds that $L \in$ cs-$\mathcal{NTIME}[n^{1+3\epsilon}, \mathrm{poly}(n)]$.*

Proof. Let L be a problem decidable by logspace-uniform $\mathcal{NC}$ circuits of size $T(n) = \mathrm{poly}(n)$ and depth $d(n)$. We use Theorem 4.2 with $T(n) = n^k$ and $d(n) = \mathrm{polylog}(n)$ and $c = 1$ and $\epsilon > 0$, in which case the cs-$\mathcal{NTIME}$ verifier's running time is at most $n^{1+3\epsilon}$. Note that $S(n) = \Theta(\log(T) \cdot (m(\bar{D}) + d))$, and without loss of generality T_P is large enough so that $\mathrm{poly}(T) \leq T_P(\bar{D})$; thus, the hypotheses of Theorem 4.2 are satisfied. ■

4.2 Refuters: How constructive is $\mathcal{NP} \neq \mathcal{PSPACE}$?

In this section we prove Theorem 1.6. Towards doing so, recall that $L \in \mathcal{PSPACE}$ is $\mathcal{PSPACE}$-complete, if for every $L_1 \in \mathcal{PSPACE}$ there is a polynomial-time computable reduction $R\colon \{0,1\}^* \to \{0,1\}^*$ such that $L_1(x) = L(R(x))$. Also recall that a language L is *paddable* if there is a polynomial-time computable function W such that for every $x \in \{0,1\}^*$ and $\ell \in \mathbb{N}$ such that

$\ell \geq |x|$, we have $L(x) = L(W(x,\ell))$ and $|W(x,\ell)| = \ell$. We remark that the standard $\mathcal{PSPACE}$-complete language TQBF is paddable.

We now prove that if $\mathcal{NP} \neq \mathcal{PSPACE}$, then there any paddable $\mathcal{PSPACE}$-complete language (in particular, TQBF) has a refuter in $\mathcal{P}^{\mathcal{NP}}$, and a refuter in FNP with $O(\log(n))$ bit of advice. The proof is an adaption of [CJS+21, Theorem 5.3].

Theorem 4.5. *Let L be a paddable $\mathcal{PSPACE}$-complete language. If $\mathcal{NP} \neq \mathcal{PSPACE}$, the following statements hold:*

1. *For every $\mathcal{NP}$ Turing machine M, there is an $\mathcal{P}^{\mathcal{NP}}$ algorithm R_M such that for infinitely many $n \in \mathbb{N}$, $R_M(1^n)$ outputs a string $x_n \in \{0,1\}^n$ such that $M(x) \neq L(x)$.*
2. *For every $\mathcal{NP}$ Turing machine M, there is an FNP algorithm R_M with $O(\log n)$ bits of advice such that for infinitely many $n \in \mathbb{N}$, $R_M(1^n)$ outputs a string $x_n \in \{0,1\}^n$ such that $M(x) \neq L(x)$.*

Proof. Let M be a polynomial-time non-deterministic Turing machine. Since $\mathcal{NP} \neq \mathcal{PSPACE}$, there are infinitely many input lengths n such that M does not correctly solve L on all n-bit inputs.

We define the language G_M as the set of $(1^n, x)$ such that $|x| \leq n$ and both of the following two statements hold:

1. There exists $y \in \{0,1\}^n$ with prefix x such that $L(y) \neq M(y)$.
2. If $|x| \geq 1$ and x ends with bit 1, then for all $y \in \{0,1\}^n$ with prefix x', $L(y) = M(y)$, where x' is the string obtained by flipping the last bit of x.

Note that $G_M \in \mathcal{PSPACE}$. We observe the following fact:

Fact 4.6. *For every $|x| < n$, either $(1^n, x \circ 0) \notin G_M$ or $(1^n, x \circ 1) \notin G_M$.*

Since L is $\mathcal{PSPACE}$-complete, there is a polynomial-time reduction $R\colon \{0,1\}^* \to \{0,1\}^*$ such that for all $n \in \mathbb{N}$ and $|x| \leq n$, we have $G_M(1^n, x) = L(R(1^n, x))$. Also since L is paddable, we can further assume that for every $|x| \leq n$, $|R(1^n, x)| = \ell(n)$, for some strictly increasing polynomial $\ell\colon \mathbb{N} \to \mathbb{N}$ such that $\ell(n) \geq n$.

We will first construct a $\mathcal{P}^{\mathcal{NP}}$ algorithm A_M that, given 1^n, outputs 3 strings x_1, x_2, x_3 with length being either n or $\ell(n)$, such that for some $i \in [3]$ we have $L(x_i) \neq M(x_i)$. We call such an algorithm a list-refuter, since it outputs a list that containing at least one bad input.[30]

From list-refuter to refuter. Before constructing such a list-refuter below, we first show that such a list-refuter A_M implies a refuter R_M, as follows: For each $i \in [3]$, we can define a refuter $R_M^{(i)}$ by letting it outputs the i-th string from the list outputted by A_M. From our assumption on A_M, we know that there exists $i \in [3]$ such that $R_M^{(i)}(1^n)$ outputs a bad input x_n for infinitely many $n \in \mathbb{N}$.

As for the output length (i.e., n vs $\ell(n)$), there are two cases: (1) There are infinitely many $n \in \mathbb{N}$, $R_M^{(i)}(1^n)$ outputs a bad input x_n and $|x_n| = n$, then $R_M^{(i)}$ is the desired refuter; (2) There are infinitely many $n \in \mathbb{N}$, $R_M^{(i)}(1^n)$ outputs a bad input x_n and $|x_n| = \ell(n)$, then we define $R_M(1^m)$ so that it simulates $R_M^{(i)}(1^n)$ where $\ell(n) = m$ (if no such n exists, $R_M(1^m)$ simply outputs 0^m). One can see that in both cases we have a standard refuter.

[30] The actual algorithm sometimes returns less than 3 strings, but we can always add dummy outputs.

- Initialize x as an empty string
- For $i \leftarrow 1, 2, \ldots, n$:
  - If $M(R(1^n, x \circ 0)) = 1$ **AND** $M(R(1^n, x \circ 1)) = 1$:
    - Return the list $R(1^n, x \circ 0)$ and $R(1^n, x \circ 1)$
  - If $M(R(1^n, x \circ 0)) = 1$:
    - $x \leftarrow x \circ 0$
  - Else if $M(R(1^n, x \circ 1)) = 1$:
    - $x \leftarrow x \circ 1$
  - Else:
    - Return the list $R(1^n, x)$, $R(1^n, x \circ 0)$, and $R(1^n, x \circ 1)$
- Return x and $R(1^n, x)$.

Figure 1: The pseudocode of the list-refuter A_M on input 1^n.

A $\mathcal{P}^{\mathcal{NP}}$ list-refuter. Our list-refuter A_M for M performs a search-to-decision reduction that extends the prefix x one bit at a time, calling $R(1^n, x)$ in each iteration. It either eventually finds a string $y \in \{0,1\}^n$ such that $L(y) \neq M(y)$, or detects the inconsistency of M's answers. See Fig. 1 for a pseudocode description of A_M.

To prove the correctness of this list-refuter, let n be an input length on which M fails to decide L. Consider four cases, according to the conditions on x when A_M terminates:

(1) $|x| < n$, $M(R(1^n, x \circ 0)) = 1$, and $M(R(1^n, x \circ 0)) = 1$. Since $(1^n, x \circ 0)$ and $(1^n, x \circ 1)$ cannot both belong to G_M by Fact 4.6, and hence either $R(1^n, x \circ 0) \notin L$ or $R(1^n, x \circ 1) \notin L$. Thus, M errs in deciding L on one of these $\ell(n)$-bit strings.

(In the rest, we assume $M(R(1^n, x \circ 0)) = 0$ or $M(R(1^n, x \circ 0)) = 0$.)

(2) $|x| = 0$. If M correctly decides L on both $R(1^n, 0)$ and $R(1^n, 1)$, then both $(1^n, 1)$ and $(1^n, 0)$ are not in G_M. This contradicts the assumption that M does not solve L correctly on all n-bit inputs.

(3) $1 \leq |x| < n$. This happens when $M(R(1^n, x)) = 1$ (from the previous iteration), but for both $b \in \{0,1\}$ it holds that $M(R(1^n, x \circ b)) = 0$. Then, it cannot be that M decides L correctly on all the outputs in the list, as that would mean that $(1^n, x) \in G_M$ but $(1^n, x \circ b) \notin G_M$ for both $b \in \{0,1\}$.

(4) $|x| = n$. This happens when $M(R(1^n, x)) = 1$ (from the previous iteration). If M is correct on $R(1^n, x)$, it means that $(1^n, x) \in G_M$, or equivalently $L(x) \neq M(x)$. Thus, M errs in deciding L on the output x.

Hence, M answers incorrectly on at least one string in the list returned by A_M.

FNP refuter. Next we show that the $\mathcal{P}^{\mathcal{NP}}$ list-refuter above can be simulated in FNP with $O(\log n)$ bits of advice on input 1^n. Applying the transformation of list-refuters to refuters, we can then get the desired FNP-refuter.

Consider the execution of $A_M(1^n)$. We first describe the advice. We use an integer $i \in [n+1]$ to indicate when in the loop the algorithm returns a list ($i = n+1$ means it returns at the end of the algorithm, after finishing the loop). We then use a Boolean bit $b \in \{0,1\}$ to indicate whether

it returns at the start of the loop ($b = 0$) or the end of the loop ($b = 1$). (If $i = n + 1$, then b does not matter.)

Then the FNP algorithm works as follows:

- It takes a string $x \in \{0,1\}^{i-1}$, together with $i - 1$ proofs $w_1, \ldots, w_{i-1}$ such that w_j is supposed to witness $M(R(1^n, x_{\leq j})) = 1$, for each $j \in [i-1]$.
- If $i = n + 1$, returns the list x and $R(1^n, x)$.
- If $b = 0$, returns the list $R(1^n, x \circ 0)$ and $R(1^n, x \circ 1)$.
- If $b = 1$, returns the list $R(1^n, x), R(1^n, x \circ 0)$, and $R(1^n, x \circ 1)$.

Given the correct advice i, it means during the execution of $A_M(1^n)$, for every $j \in [i-1]$, it holds that in the j-th stage of the loop, exactly one of $M(R(1^n, x \circ 0)) = 1$ and $M(R(1^n, x \circ 1)) = 1$ holds. This in particular implies exactly one string $x \in \{0,1\}^{i-1}$ admits proofs $w_1, \ldots, w_{i-1}$. The correctness of the simulation then follows from the definition of advice b. ■

4.3 Bounded arithmetic: How provable is $\mathcal{NP} \neq \mathcal{PSPACE}$?

In this section we prove Theorem 1.5. For this purpose, we will assume basic familiarity with the main notions in the area of bounded arithmetic. Detailed definition can be found in textbooks such as the ones by Krajíček [Kra95; Kra19] and by Cook and Nguyen [CN10], and a recent gentle introduction for theorists who are not familiar with the area can be found in [LO23]. (The latter work focuses, in particular, on the main theory that we refer to, which is Jeřábek's APC_1.)

Formulation. Let M be an $O(n)$-space Turing machine, and $V(x, y)$ be a polynomial-time verifier that takes input $x \in \{0,1\}^n$ and $y \in \{0,1\}^{p(n)}$ for some polynomial $p \colon \mathbb{N} \to \mathbb{N}$. The following formula $\Phi_{M,V}$ asserts that V is an $\mathcal{NP}$-verifier for the language decided by M:

$$\begin{aligned}\Phi_{M,V} \coloneqq \exists n_0 \in \mathsf{LogLog}\ \forall n \in \mathsf{LogLog}\ \text{s.t. } n > n_0\ \forall x \in \{0,1\}^n \\ (M(x) = 1 \implies \exists y \in \{0,1\}^{p(n)} V(x, y) = 1) \\ \wedge (M(x) = 0 \implies \forall y \in \{0,1\}^{p(n)} V(x, y) = 0).\end{aligned}$$

The $n \in \mathsf{LogLog}$ notation means that $\exists N \exists n = ||N||$, where $|\cdot|$ denotes the length of the binary string corresponding to an integer. In particular, since the formula is actually defined with respect to input length N, it allows the inside formula to reason about $2^{O(n)}$-time computation. Since $M(x)$ can be evaluated in time $2^{O(n)}$, the formula $\Phi_{M,V}$ can be formulated as a PV_1 sentence.[31]

We remark that $(M(x) = 1 \implies \exists y \in \{0,1\}^{p(n)} V(x, y) = 1)$ states the *completeness* of the proof system V, and $(M(x) = 0 \implies \forall y \in \{0,1\}^{p(n)} V(x, y) = 0)$ states the *soundness* of the proof system V. The whole formula $\Phi_{M,V}$ says that for all sufficiently long input x, $M(x) = 1$ if and only if there is a proof $y \in \{0,1\}^{p(n)}$ such that $V(x, y) = 1$.

[31] Let us provide more detail on how we construct $\Phi_{M,V}$ as a PV_1 sentence. We define a PV_1 function $U(M, x, 1^s, 1^t)$ that simulates a Turing machine M using at most s bits of space and t steps, on a given input x; if the simulation requires more than s bits of space or t steps of time, U outputs $\perp$. In the formula $\Phi_{M,V}$, the symbol $M(x)$ is just an abbreviation for $U(M, x, c_M n, 2^{c_M n})$, where $c_M \in \mathbb{N}$ is a constant depending on M.

We first recall the KPT witnessing theorem for APC_1, from [CKK+21]. Loosely speaking, the result asserts that if the theory APC_1 proves that for every $n \in \mathbb{N}$ there exists x such that $\forall z, \varphi_n(x,z)$, then given n we can *efficiently* find x such that $\forall z, \varphi_n(x,z)$. In the statement below, the notation $\forall n \in \mathsf{Log}$ means $\forall N \forall n = |N|$.

Theorem 4.7 (KPT witnessing for APC_1; see [CKK+21, Theorem 4.7])**.** *Let φ be a $\Sigma^B_0(\mathsf{PV}_1)$ formula such that*

$$\mathsf{APC}_1 \vdash \forall n \in \mathsf{Log}\, \exists C\, \forall Z :\ \varphi(n, C, Z)\ .$$

Then there is a constant number ℓ of $\mathrm{poly}(n)$-time computable functions

$$A_1(n, R_1), A_2(n, R_1, Z_1, R_2), \ldots, A_\ell(n, R_1, Z_1, \ldots, R_{\ell-1}, Z_{\ell-1}, R_\ell)$$

and a constant $c \geq 1$ such that, for every $n \geq 1$, the following holds.

1. *With probability at least $1/n^c$ over uniform randomness R_1, for $C_1 = A_1(n, R_1)$, either it holds that $\forall Z_1 \varphi(n, C_1, Z_1)$, or for any Z_1 such that $\neg\varphi(n, C_1, Z_1)$ holds, the following holds.*
2. *With probability at least $1/n^c$ over uniform randomness R_2, for $C_2 = A_1(n, R_1, Z_1, R_2)$, either it holds that $\forall Z_2 \varphi(n, C_2, Z_2)$, or for any Z_2 such that $\neg\varphi(n, C_2, Z_2)$ holds, the following holds.*

$$\vdots$$

3. *With probability at least $1/n^c$ over uniform randomness R_ℓ, for $C_\ell = A_\ell(n, R_1, Z_1, \ldots, R_{\ell-1}, Z_{\ell-1}, R_\ell)$, we have $\forall Z_\ell \varphi(n, C_\ell, Z_\ell)$.*

We now state Theorem 1.5 formally and prove it, relying on Theorems 1.3 and 4.7. This consequence of Theorem 1.3, as well as the proof approach below, were suggested by Ján Pich.

Theorem 4.8 ($\mathsf{APC}_1 \nvdash \mathcal{NP} \neq \mathcal{PSPACE}$)**.** *Suppose that the assumptions of Theorem 1.3 hold. Let M be a $O(n)$-space Turing machine deciding a language $L = L_M$, let V be the polynomial-time $\mathtt{cs}$-$\mathcal{NTIME}$ verifier for L from Theorem 1.3, and let p be a polynomial bounding the proof length of V. Then,*

$$\mathsf{APC}_1 \nvdash \neg\Phi_{M,V}\ ;$$

in other words, there is a model of APC_1 in which $\Phi_{M,V}$ holds.

Proof. For the sake of contradiction, assume that

$$\mathsf{APC}_1 \vdash \neg\Phi_{M,V}.$$

Equivalently, we have that

$$\begin{aligned}\mathsf{APC}_1 \vdash \forall n_0 \in \mathsf{LogLog}\, \exists n \in \mathsf{LogLog}\ \text{s.t.}\ n > n_0 \\ \exists x \in \{0,1\}^n \exists y \in \{0,1\}^{p(n)} \forall z \in \{0,1\}^{p(n)} \\ (M(x) = 0 \wedge V(x,y) = 1) \vee (M(x) = 1 \wedge V(x,z) = 0).\end{aligned}$$

Recall that, as explained in the beginning of this section, $\Phi_{M,V}$ is actually defined using $\exists N \in \mathsf{Log}, n_0 = |N|$ (i.e., we used $\exists n_0 \in \mathsf{LogLog}$ in the definition of $\Phi_{M,V}$ only as shorthand). Thus, we can apply Theorem 4.7 to $\neg\Phi_{M,V}$, resulting in algorithms with running time $\mathrm{poly}(N) = 2^{O(n_0)}$:

$$A_1(n_0, R_1), A_2(n_0, R_1, Z_1, R_2), \ldots, A_\ell(n_0, R_1, Z_1, \ldots, R_{\ell-1}, Z_{\ell-1}, R_\ell).$$

Also note that, denoting by φ the Σ_0^B part of $\neg\Phi_{M,V}$, we have

$$\varphi(n_0, C, Z) = \varphi(n_0, (n, x, y), z) := (M(x) = 0 \wedge V(x, y) = 1) \vee (M(x) = 1 \wedge V(x, z) = 0).$$

We then describe how to convert the algorithms $A_1, ..., A_\ell$ into a cheating prover A for V that contradicts the soundness guaranteed by Theorem 1.3.

The prover A works in ℓ stages. For $i = 1, ..., \ell$, the i^{th} stage is conducted as follows.

The input to the i^{th} stage of A is $(n_{i-1}, z_1, ..., z_{i-1})$ (in the first stage $i = 1$ the input is just n_0). Drawing R_i uniformly at random, A computes $C_i = (n_i, x_i, y_i) = A_i(n_0, R_1, z_1, \ldots, R_{i-1}, z_{i-1}, R_i)$ (when $i = 1$, A just computes $A_1(n_0, R_1)$).

If $i = \ell$, then A outputs (x_ℓ, y_ℓ).

If $M(x_i) = 0$ and $V(x_i, y_i) = 1$, then A outputs (x_i, y_i) and halts.

Otherwise, the goal of A is to find a z_i such that $\neg\varphi(n_{i-1}, (n_i, x_i, y_i), z_i)$ holds (to feed into A_{i+1} in the next stage). By the definition of φ, we have

$$\begin{aligned}\neg\varphi(n_{i-1}, (n_i, x_i, y_i), z) &= (M(x_i) = 1 \vee V(x_i, y_i) = 0) \wedge (M(x_i) = 0 \vee V(x_i, z) = 1) \\ &= (M(x_i) = 0 \implies V(x_i, y_i) = 0) \wedge (M(x_i) = 1 \implies V(x_i, z) = 1)\end{aligned}$$

There are two cases:

- If $M(x_i) = 0$, then by our assumption $V(x_i, y_i) = 0$. Hence, any $z = z_i$ works.
- Otherwise $M(x_i) = 1$. Let $c = c_L$ be the constant when applying Theorem 1.3 to the language L. By the existence of an honest prover for L with running time 2^{n^c}, we can compute z_i such that $V(x_i, z_i) = 1$ in time $2^{O(n_i^c)}$, and we continue with this z_i (note that here we do not need the completeness to be formalized in APC_1).

We claim that, with noticeable probability, A produces (x, y) such that $M(x) = 0$ but $V(x, y) = 1$. In more detail:

Fact 4.8.1. *For all $n_0 \in \mathbb{N}$, given 1^{n_0} as input, A outputs an integer $n \geq n_0$, $x \in \{0,1\}^n$, and $y \in \{0,1\}^{p(n)}$ such that $M(x) = 0$ and $V(x, y) = 1$ with probability at least $2^{-O(n_0)}$, in time $2^{O(n^c)}$.*

Proof. Note that A either outputs (x_i, y_i) such that $M(x_i) = 0$ and $V(x_i, y_i) = 1$, or outputs (x_ℓ, y_ℓ). By Theorem 4.7, in the latter case, with probability at least $2^{-O_\ell(n_0)}$, for all z we have that

$$(M(x_\ell) = 0 \wedge V(x_\ell, y_\ell) = 1) \vee (M(x_\ell) = 1 \wedge V(x_\ell, z) = 0)\ .$$

However, since for every x_ℓ such that $M(x_\ell) = 1$ there is z such that $V(x_\ell, z) = 1$ (we know this since the honest prover produces such a z), it must be that $M(x_\ell) = 0$ and $V(x_\ell, y_\ell) = 1$. □

We say that an integer $n \in \mathbb{N}$ is good if there exists $n_0 \in [n]$ such that $A(1^{n_0})$ outputs n, $x \in \{0,1\}^n$, and $y \in \{0,1\}^{p(n)}$ satisfying the properties above with probability at least $2^{-O(n_0)}$. From the properties of A, we know that every $n_0 \in \mathbb{N}$ contributes to at least one good $n \geq n_0$, hence there are infinitely many good n. Also note that $2^{-O(n_0)} \geq 2^{-O(n)}$ since $n_0 \leq n$.

Let κ_0 be the constant in the big-O of $2^{O(n^c)}$ from Fact 4.8.1. We can then construct a cheating prover P from A as follows: Given input 1^n, try all $n_0 \in [n]$ (from 1 to n) and see if running $A(1^{n_0})$ for $2^{\kappa_0 \cdot n^c}$ steps outputs a pair $x \in \{0,1\}^n$ and $y \in \{0,1\}^{p(n)}$ such that $M(x) = 0$ and $V(x,y) = 1$; if so, it outputs x and y. Now, for those infinitely many good $n \in \mathbb{N}$, it holds that $P(1^n)$, in $2^{O(n^c)}$ time, outputs a pair $x \in \{0,1\}^n$ and $y \in \{0,1\}^{p(n)}$ such that $M(x) = 0$ and $V(x,y) = 1$ with probability at least $2^{-O(n)}$; this contradicts Theorem 1.3. ■

5 Zaps, NIWI and NIZK

In Section 5.1 we prove the results that were stated in Section 1.3; that is, Theorem 1.8 and Corollary 1.11. In Section 5.2 we expand on the exposition in Section 1.3 by surveying prior works in more detail.

5.1 Proof of Theorem 1.8

Let us restate Theorem 1.8 and prove it.

Theorem 5.1. *Assume that there are subexponentially secure one way functions. Let $\delta > 0$ be a sufficiently small constant,[32] and let $\epsilon > 0$ be an arbitrary constant. Assume that there is $f\colon \{0,1\}^n \to \{0,1\}^{n^\delta}$ in $\mathcal{FP}$ that is hard for linear-time algorithms with oracle access to $\mathcal{NTIME}[n^\epsilon]$ over all polynomial-time samplable distributions. Then, every $\mathcal{NP}$ relation R has a zap argument system.*

Assuming in addition a sub-exponentially secure non-interactive commitment scheme (i.e., statistically binding and computationally hiding), every $\mathcal{NP}$ relation R has a NIWI *argument system.*

Recall that, combining the concluded zap in Theorem 5.1 with Lemma 3.7, we obtain NIZK arguments for $\mathcal{NP}$; thus, Corollary 1.11 follows from Theorem 5.1. The rest of this section is thus devoted to the proof of Theorem 5.1.

We focus on building an argument system for the $\mathcal{NP}$-complete problem of Hamiltonicity. In the following, we will refer to a commitment scheme that is statistically binding and computationally hiding. In case the assumption is a one-way function, this commitment scheme is Naor's [Nao91] construction, which uses a CRS; in case the assumption is a non-interactive commitment scheme, that will be the scheme we are referring to. For simplicity of presentation, we will describe the constructions of the zap and of the NIWI together, since the only difference between them is that in the former a CRS used to instantiate the commitment scheme.

Blum's classical protocol for Hamiltonicity. Both parties are given a graph G over n vertices, represented by its adjacency matrix; when G is Hamiltonian, the honest prover gets a a Hamiltonian cycle w in G. The parties repeat the following interaction $t = \text{poly}(n)$ times in parallel:

1. The prover commits to a permutation π of $[n]$ and to the edges of $\pi(G)$.[33]

[32] Specifically, we need $\delta > 0$ to be smaller than a constant that is determined by the subexponential security of the one-way function.

[33] That is, for each $u, v \in [n]$, the prover commits to a bit $e_{u,v}$ indicating whether or not u and v are neighbors.

2. The verifier sends a random bit $b \in \{0,1\}$.

3. If $b = 0$ the prover decommits to π and to $\pi(G)$, and the verifier checks that the permutation sent indeed maps the original graph G to the graph sent.

   If $b = 1$, the prover decommits to the edges of the Hamiltonian cycle in $\pi(G)$, and the verifier checks that these edges indeed yield a Hamiltonian cycle.

We assume that t is a sufficiently large polynomial, depending on $\epsilon > 0$. The commitment is instantiated with security parameter $\lambda = \text{poly}(n)$ such that it is secure against time $2^{O(t)} = 2^{\lambda^{\Omega(1)}}$. We denote the t initial commitments by $\vec{c} = c_1, ..., c_t$ and the t response bits by $\vec{b} = b_1, ..., b_t$. Note that $|\vec{c}| = O(t \cdot n^2 \cdot \text{poly}(\lambda)) = \text{poly}(n)$ and $|\vec{b}| = t$.

The derandomized protocol. On a given G, the honest prover computes $\vec{c}$ identically to Blum's protocol, and then its goal is to compute $\vec{b}$ by applying f to input $(G, \vec{c})$ (see below). Then, it computes its final response $\vec{m}$ as in Blum's protocol and sends $(\vec{c}, \vec{b}, \vec{m})$ to the verifier. The verifier checks that $\vec{b}$ is the result of applying f to $(G, \vec{c})$, and then performs the same verification as in Blum's protocol with respect to the fixed choice $\vec{b}$.

We denote the resulting prover and verifier by P and V. Since $f \in \mathcal{FP}$, the verifier V runs in polynomial time, and the honest prover P runs in polynomial time given the witness w.

A minor technical issue is that the length of $(G, \vec{c})$ needs to be such that f applied to it yields exactly t bits. To handle this, observe that

$$|f(G, \vec{c})| \leq t^{\delta} \cdot n^{2\delta} \cdot \lambda^{O(\delta)} < t^{2\delta/\zeta} \cdot n^{2\delta} \,,$$

where $\zeta > 0$ is a constant that comes from the quality of the commitment scheme.[34] Since we assumed that $\delta > 0$ is sufficiently small, it follows that $|(G, \vec{c})|^{\delta} < t$. The prover and verifier pad $(G, \vec{c})$ with zeroes to be of length $t^{1/\delta}$ and apply f to the padded version to obtain $\vec{b} \in \{0,1\}^t$.

Completeness and soundness. Completeness follows since Blum's protocol has perfect completeness. For soundness, we analyze the protocol conditioned on the event that the commitment is perfectly binding. (This is always true in the case of a non-interactive commitment scheme, and is true with all but negligible probability over choice of CRS when using [Nao91].)

Given a non-Hamiltonian G and any fixed $\vec{c}$, let $B = B(G, \vec{c}) \subseteq \{0,1\}^t$ be the set of messages $\vec{b}$ such that there exists a final message $\vec{m}$ by the prover causing the verifier to accept.[35] As observed in [CCH+19], the set B is actually a singleton:

Fact 5.2. *For every non-Hamiltonian G and fixed $\vec{c}$, there is at most one $\vec{b}$ such that there exists $\vec{m}$ causing the verifier to accept. Furthermore, the function mapping $(G, \vec{c})$ to the unique $\vec{b}$ (whenever $\vec{b}$ exists) is computable in linear time with oracle access to $\mathcal{NTIME}[\text{poly}(\lambda)] = \mathcal{NTIME}[N^{\epsilon}]$, where $N = |(G, \vec{c})|$ is the input length to this function.*

Proof. Fix $i \in [t]$, let G' be the graph that the prover committed to in c_i, and let π' be the permutation that the prover committed to in c_i. (Recall that c_i is statistically binding, and so π' is uniquely determined by c_i.)

[34] Recall that we set λ to be such that the scheme is secure against time $2^{O(t)} = 2^{\lambda^{\zeta}}$.

[35] That is, the verifier accepts the transcript $(G, \vec{c}, \vec{b}, \vec{m})$.

- If $G' = \pi'(G)$, then the verifier can accept only if $b_i = 0$ (because G' doesn't have a Hamiltonian cycle, so the verifier will never accept with $b_i = 1$).
- If $G' \neq \pi'(G)$, the verifier can accept only if $b_i = 1$ and G' is Hamiltonian.

Since the above holds for every $i \in [t]$, we deduce that B has at most one element. To see that B is computable as claimed, note that it suffices to check, for every i, if the prover committed in c_i to G' and π' such that $G' = \pi'(G)$ (if so, then we set $b_i = 0$, and otherwise we set $b_i = 1$). This can be done by breaking the commitment c_i, which can be done using the $\mathcal{NTIME}[\lambda^k]$ oracle, where $k > 1$ is a constant that depends on the commitment scheme.

Relying on t being a sufficiently large polynomial (as a function of ϵ and of k), we have that $|(G, \vec{c})| = O(t \cdot n^2 \cdot \text{poly}(\lambda)) \geq \lambda^{k/\epsilon}$, and hence the oracle is computable in non-deterministic time $|(G, \vec{c})|^\epsilon$. □

Now, assume towards a contradiction that there is a ppt algorithm $\tilde{P}$ such that with noticeable probability over $(G, \pi) \leftarrow \tilde{P}(1^n)$ the verifier accepts G with proof π but G is not Hamiltonian. We say that (G, π) is bad in such a case, and denote $\pi = (\vec{c}, \vec{b}, \vec{m})$.

By Fact 5.2, for every bad (G, π) it holds that $\vec{b} = f(G, \vec{c})$ is the unique string such that the verifier accepts the transcript. Hence, whenever (G, π) is bad the linear-time algorithm with $\mathcal{NTIME}[n^\epsilon]$ oracle that computes this string (from Fact 5.2) succeeds in computing f on input $(G, \vec{c})$. This contradicts the hardness of f.

Zero knowledge. Our goal now is to prove that the construction is an unbounded simulator zap; that is, we show a computationally unbounded simulator S satisfying the condition in Definition 3.4. Note that the WI zap property will follow from this fact, using Lemma 3.6.

On input (G, crs), the simulator S repeats the following procedure at most $r = 2^{O(t)}$ times.

1. Select a random $\vec{b}_{sim} = (b_1, ..., b_t) \in \{0,1\}^t$.
2. Generate $\vec{c}$ as follows. For every $i \in [t]$, if $b_i = 0$ then c_i is a commitment to a random permutation π_i and to the graph $\pi_i(G)$; and if $b_i = 1$ then commit to the all-zero permutation π_i and to a graph that is a random cycle on n vertices.
3. Compute f on the padded version of $(G, \vec{c})$.

   If the result is different from $\vec{b}_{sim}$, continue to the next attempt.

   Otherwise, compute the following decommitments $\vec{m}$: For each $i \in [t]$, if $b_i = 0$ decommit to π_i and to $\pi_i(G)$; and if $b_i = 1$ decommit to the edges in the cycle. Output the transcript $(\vec{c}, \vec{b}_{sim}, \vec{m})$.

In case all r iterations fail, output $\perp$.

Notation and basic facts. Let crs be a common random string, G an input graph, w a Hamiltonian path in G, and z an auxiliary input (jumping ahead, we will be considering (crs, G, w, z) generated by an adversary as in Definition 3.4 but for the moment we view them as fixed). For these fixed (crs, G, w, z), we define a sequence of random variables that will be useful for our analysis.

- $\xi_{real}(crs, G, w, z)$. Select a random $\vec{b} \in \{0,1\}^t$. For each $i \in [t]$ independently, generate a random permutation π_i and commit to $(\pi_i, \pi_i(G))$ (the commitment uses fresh random coins for each i). Let $\vec{c}$ be the sequence of t commitments. Compute f on the padded version of $(G, \vec{c})$; if the result is not $\vec{b}$ then output $\perp$, otherwise output $(crs, G, (\vec{c}, \vec{b}, \vec{m}), z)$ where $\vec{m}$ is the honest prover's decommitments.

- $\xi_{mid}(crs, G, w, z)$: The only difference between this RV and ξ_{real} is $\vec{c}$. For every $i \in [t]$, if $b_i = 0$ then c_i is without change; but if $b_i = 1$ then c_i is a commitment to the identity permutation and to a graph that *only* has the cycle $\pi_i(w)$ (and the rest of the edges are removed). For convenience, we assume that the identity permutation is represented by the all-zero string.[36]

- $\xi_{sim}(crs, G, w, z)$: Again, the only difference between this RV and ξ_{mid} is $\vec{c}$ in blocks i where $b_i = 1$. Now c_i is a commitment to the identity permutation and to a random cycle on n vertices.

- $\rho_{real}(crs, G, w, z)$: Take r independent copies of $\xi_{real}(crs, G, w, z)$ and output the first output that is not $\perp$. In case all copies are $\perp$ then output $\perp$.

- $\rho_{mid}(crs, G, w, z)$: Is defined similarly to $\rho_{real}(crs, G, w, z)$ relative to ξ_{mid}.

- $\rho_{sim}(crs, G, w, z)$: Is defined similarly to $\rho_{real}(crs, G, w, z)$ relative to ξ_{sim}.

We now observe that ρ_{real} is (up to a small error) the real distribution that the verifier sees, whereas ρ_{sim} is the distribution simulated by S. We also observe that ξ_{mid} and ξ_{sim} are identically distributed. Jumping ahead, we will later use these facts to show that an adversary distinguishing the real interaction ρ_{real} from the simulated one ρ_{sim} also distinguishes ξ_{real} from ξ_{mid}; since the latter two only differ in parts of the commitment that are never exposed in the interaction, such an adversary breaks the commitment.

Claim 5.3. *For any (crs, G, w, z), the RV $\rho_{real}(crs, G, w, z)$ is $2^{-\omega(t)}$-close (in statistical distance) to the RV $P(crs, G, w)$ (i.e., to the view of the verifier on input G and CRS crs, when the prover gets the cycle w).*

Proof. Conditioned on not being equal to $\perp$, the RV $\xi_{real}(crs, G, w, z)$ is distributed identically to the view of the verifier. The claim now follows by noting that the probability that $\xi_{real}(crs, G, w, z) \neq \perp$ is 2^{-t} (since this is the probability that a random $\vec{b}$ equals the output of f on $(G, \vec{c})$, and the random coins used to generate the commitments $\vec{c}$ are independent of the random coins used to generate $\vec{b}$), and the experiment underlying ρ_{real} is repeated $r = 2^{O(t)}$ times. □

Claim 5.4. *For any (crs, G, w, z), the RV $\rho_{sim}(crs, G, w, z)$ is distributed identically to the RV $(crs, G, S(G, crs))$.*

Proof. Follows immediately from the definitions of ρ_{sim} and of S. □

Claim 5.5. *For any (crs, G, w, z), the RV $\xi_{sim}(crs, G, w, z)$ is distributed identically to the RV $\xi_{mid}(crs, G, w, z)$.*

Proof. The only difference between the definitions of the RVs is that in ξ_{mid} the commitment is to a random permutation of a fixed cycle and in ξ_{sim} the commitment is to a random cycle. □

[36] A natural representation in which this happens is when permutations are represented by specifying, for each $i \in [n]$, the shift of the bits of i that the permutation induces.

The security reduction. Let (A_1, A_2) be a pair of ppt adversaries for the unbounded simulator property. Recall that A_1 on input 1^n outputs (crs, G, w, z), and A_2 on input (crs, G, α, z) attempts to decide whether α is generated by the prover or by the simulator. We assume without loss of generality that for any (G, w) generated by A_1 it holds that $(G, w) \in R$ (since otherwise A_1 can just abort without changing the overall advantage of (A_1, A_2)). We also assume without loss of generality that z (which is passed from A_1 to A_2) incorporates the witness w.[37]

In what follows, we use (crs, G, w, z) to denote the **distribution generated by** $A_1(1^n)$. We stress this because in the definitions and claims above, (crs, G, w, z) was fixed, whereas from now on it will be distributed according to $A_1(1^n)$.

Assume toward a contradiction that there exists a polynomial p such that

$$\left|\Pr\left[A_2\big(crs, G, P(crs, G, w), z\big) = 1\right] - \Pr\left[A_2\big(crs, G, S(crs, G), z\big) = 1\right]\right| \geq 1/p(n)\ , \tag{5.1}$$

for infinitely many n. We first argue that there is an efficient adversary distinguishing ξ_{real} from ξ_{sim}; the claim amounts to a hybrid argument, but when working in the uniform setting this argument needs to be carried out more carefully.

Claim 5.6. *There exists a* $\mathrm{poly}(r)$*-time algorithm* D' *such that:*

$$\left|\Pr\left[D'\big(crs, G, \xi_{real}(crs, G, w, z), z\big) = 1\right] - \Pr\left[D'\big(crs, G, \xi_{sim}(crs, G, w, z), z\big) = 1\right]\right| \geq \frac{1}{2r \cdot p(n)},$$

for infinitely many n*.*

Proof. Combining Eq. (5.1) with Claims 5.3 and 5.4 (which hold point-wise for every fixed (crs, G, w, z)), we have that

$$\left|\Pr\left[A_2\big(crs, G, \rho_{real}(crs, G, w, z), z\big) = 1\right] - \Pr\left[A_2\big(crs, G, \rho_{sim}(crs, G, w, z), z\big) = 1\right]\right| \geq 1/2p(n)\ . \tag{5.2}$$

For any *fixed* (crs, G, w, z) we define a sequence of hybrid distributions $H_0, ..., H_r$. The distribution H_0 is sampled by independently sampling r instances of ξ_{sim}, and outputting the first instance that is not $\perp$ (or outputting $\perp$ if there is no such instance). For $i \in [r]$ the distribution H_i is sampled by sampling i independent instances of ξ_{real} and $r - i$ independent instances of ξ_{sim}, and outputting the first instance among the r that is not $\perp$ (or $\perp$ if there is no such instance).

Note that H_0 is precisely ρ_{sim} and H_r is precisely ρ_{real}. Thus, Eq. (5.2) can be presented as

$$\left|\Pr\left[A_2\big(crs, G, H_r, z\big) = 1\right] - \Pr\left[A_2\big(crs, G, H_0, z\big) = 1\right]\right| \geq 1/2p(n)\ ,$$

and hence there is $\sigma \in \{0, 1\}$ such that for infinitely many n,

$$\mathbb{E}_{i \in [r]}\left[\Pr[A_2(crs, G, H_i, z) = \sigma] - \Pr[A_2(crs, G, H_{i-1}, z) = \sigma]\right] \geq \frac{1}{2r \cdot p(n)}\ ,$$

where the internal probability is also over choice of (crs, G, w, z) by A_1.

[37] We will indeed use this assumption in the proof below: The distinguisher that breaks the commitment scheme will have access to the generated auxiliary information z, which incorporates w. We comment, though, that incorporating w into z is not crucial for this purpose, since the distinguisher can also have access to w itself. The only crucial point is that the distinguisher can generate (G, w) efficiently (i.e., using A_1) when trying to break the commitment.

The algorithm D'_σ gets (crs, G, z, α) and creates a string β as follows. It chooses $i \in [r]$ uniformly at random, then generates $i-1$ independent instances of ξ_{real} and $r-(i-1)$ independent instances of ξ_{sim}. Note that to generate ξ_{real} it needs a witness w, which it gets from z (recall that we assumed wlog that A_1 includes w in z, and thus for our purposes we can define D'_σ that expects w in z). The string β is the first component among the r that is not $\perp$ (or $\perp$ if all are $\perp$).

Observe that for any fixed choice of (crs, G, w, z) by A_1, if D'_σ gets the corresponding RV (crs, G, z, ξ_{sim}) then the distribution of β is identical to H_{i-1}, and if D'_σ gets the RV (crs, G, z, ξ_{real}) then β is distributed identically to H_i.

The algorithm D'_σ feeds (crs, G, β, z) into A_2, and outputs $A_2(crs, G, \beta, z) \oplus \sigma$. By the above, for some σ this algorithm has distinguishing advantage of at least $1/(2r \cdot p(n))$. □

Combining Claim 5.6 with Claim 5.5, we deduce that D' distinguishes between ξ_{real} and ξ_{mid} (i.e., rather than just between ξ_{real} and ξ_{sim}); that is:

$$\left|\Pr\left[D'\left(crs, G, \xi_{real}(crs, G, w, z), z\right) = 1\right] - \Pr\left[D'\left(crs, G, \xi_{mid}(crs, G, w, z), z\right) = 1\right]\right| \geq \frac{1}{2r \cdot p(n)},$$

for infinitely many n.

We now define an adversary D'' that distinguishes u commitments to zeroes from u commitments to ones, where u is the number of bits the honest prover commits to in $\vec{c}$; that is, $u = O(t \cdot (n \cdot \log n + n^2)) = O(t \cdot n^2)$. The adversary D'' uses A_1 to generate (crs, G, w, z). It then chooses t random permutations $\pi_1, ..., \pi_t$ and a random $\vec{b} \in \{0,1\}^t$, and for each $i \in [t]$ it generates a "commitment" c_i to $(\pi_i, \pi_i(G))$ in the following way.

1. If $b_i = 0$, then it uses *crs* and fresh random coins to generate a (true) commitment to $(\pi_i, \pi_i(G))$.
2. If $b_i = 1$, then for each bit of $(\pi_i, \pi_i(G))$, if the bit is zero then it uses *crs* and fresh random coins to generate a true commitment to that bit, and if the bit is one then it uses a commitment that it received as input.[38] However, one exception is that for the bits of the cycle $\pi_i(w)$, it uses *crs* and random coins to generate true commitments.

Denote the commitments above by $\tilde{c}$. If the value of f on the padded version of $(G, \tilde{c})$ is not $\vec{b}$, then D'' outputs $\perp$. Otherwise, it outputs $(crs, G, (\vec{b}, \tilde{c}, \vec{m}), z)$ where $\vec{m}$ is the honest prover's response when choosing permutation π and receiving challenge $\vec{b}$ from the verifier. Observe that:

Claim 5.7. *When the input to D'' is commitments to one, the output distribution of D'' is identical to (crs, G, ξ_{real}, z). When the input is commitments to zero, the output distribution of D'' is identical to (crs, G, ξ_{mid}, z).*

Proof. Recall that the only difference between ξ_{real} and ξ_{mid} is in commitments c_i for indices $i \in [t]$ where $b_i = 1$: In ξ_{real} the commitments in c_i are for the bits of $(\pi_i, \pi_i(G))$, whereas in ξ_{mid} these are commitments to a string that has zeroes everywhere except the cycle $\pi_i(w)$ (and in these locations it indeed has commitments to the bits of $\pi_i(w)$).

Fix a choice of $\vec{b} \in [t]$, and fix $i \in [t]$. If $b_i = 0$ then D'' generates c_i identically to the way c_i is generated in ξ_{real} and in ξ_{mid} (conditioned on the fixed choice of $\vec{b}, i$). If $b_i = 1$, then the

[38] Each input commitment is used at most once, for some $i \in [t]$ and for some bit of $(\pi_i, \pi_i(G))$.

following holds: When the input to D'' is commitments to zero, the output is a commitment to the all-zero string except for the locations in $\pi_i(w)$, which is identical to ξ_{mid}; and when the input is commitments to one, the output is a commitment to $(\pi_i, \pi_i(G))$, which is identical to ξ_{real}. □

By combining D'' with D' from Claim 5.6, we get an algorithm D that distinguishes between the two cases in time $\text{poly}(r) = 2^{O(t)}$ with advantage $\frac{1}{2r \cdot p(n)}$, for infinitely many n.

The last step is another hybrid argument. Consider $u+1$ hybrid distributions $H_0, ..., H_u$, where H_i consists of i commitments to zero and $u-i$ commitments to one. Since $|\Pr[D(H_0) = 1] - \Pr[D(H_u) = 1]| \geq \frac{1}{2r \cdot p(n)}$, for some σ it holds that

$$\mathbb{E}_{i \in [u]} \left[\Pr[D(H_i) = \sigma] - \Pr[D(H_{i-1}) = \sigma] \right] \geq \frac{1}{r \cdot u \cdot p(n)}$$

for infinitely many n.

The final adversary distinguishes between a commitment to zero and a commitment to one as follows. It chooses $i \in [u]$ uniformly at random, creates u commitments by placing its received commitment at location i and generating the other commitments accordingly,[39] and calling D on the generated commitments. This algorithm runs in time $\text{poly}(r)$ and achieves advantage $1/\text{poly}(r)$, contradicting the security of the commitment scheme.

Having established the completeness, soundness and unbounded simulator property, we deduce that the proof system is a zap, and also a NIWI in case the commitment is in the plain model. This concludes the proof of Theorem 5.1.

5.2 Prior work on zaps, NIWI and NIZK

Prior to our work there were only three main constructions of NIWIs, all from highly structured assumptions, as we describe next.

- Barak, Ong and Vadhan [OV07] gave the first construction of a NIWI. Their construction relies on a combination of two assumptions: (1) statistically-sound zaps (in the uniform common random string model) and (2) a NW-style derandomization assumption. Statically sound zaps are known from (1) (doubly-efficient) trapdoor permutations [DN07] (which in turn are only known based on factoring [RSA78; Rab79] or from indistinguishability obfuscation (IO) [BPW16], which is discussed below), (2) the decisional linear assumption on groups equipped with a bilinear map [GOS06] or (3) directly from IO [BP15].

- Groth, Ostrovsky and Sahai [GOS06] directly constructed NIWI based on the decisional linear assumption on groups equipped with a bilinear map.

- Bitansky and Paneth [BP15] construct NIWI from IO and one-way permutations. While by now IO is known to exist based on fairly standard assumptions [JLS21; JLS22], the assumptions are still very specific and highly structured.

Needless to say, the above constructions of NIWI are all also zaps. In addition to these, we mention the works of [BFJ+20; LVW19; LVW20; GJJ+20; CKS+21] who construct statistical zap

[39] That is, the first $i-1$ commitments are commitments to zero and the last $u-i$ commitments are commitments to one.

arguments (some of which are only privately-verifiable), from problems such as learning with errors or problems related to discrete log.

As for NIZKs, for a long time the only known constructions relied on factoring [FLS90] or hardness assumptions on bilinear groups [GOS06]. Recently however, the aforementioned thread on secure instantiations of Fiat-Shamir have led to constructions relying on LWE [CCH+19; PS19], Diffie-Hellman style-assumptions [JJ21; CJJ+23] or certain combinations of LPN with other assumptions [BKM20].

6 Correlation intractability vs derandomization

Our goal in this section is to prove an equivalence between certain targeted hitting-set generators (targeted HSGs), and a relaxed variant of correlation intractable hash functions (CIHFs). This equivalence sheds new light on both objects, and more generally on the connection between correlation intractability, Fiat-Shamir, and derandomization.

Working in the uniform setting. The focus of this work is the setting of *uniform algorithms*. Thus, we will consider targeted generators that are pseudorandom for uniform algorithms, and CIHFs that are secure against relations recognizable by uniform algorithms.

We stress that we do not know how to securely instantiate the Fiat-Shamir heuristic for interesting classes of proof systems when the class of relations is (say) all relations recognizable by uniform algorithms running in some fixed polynomial time. (As an illustrative example, note that the relation underlying the proof of [HL18] is recognizable only by *non-uniform* circuits.) Nevertheless, one may view the current work as demonstrating that Fiat-Shamir can be securely instantiated for the GKR proof system with CIHFs that avoid relations recognizable by uniform *small-space* algorithms (as in Theorem 1.3 and Corollary 7.6).

Comparing "apples-to-apples". Recall that a standard CIHFs is a keyed collection of functions $\{h_\alpha : \{0,1\}^n \to \{0,1\}^m\}_\alpha$. Using standard CIHFs that avoid relations from a sufficiently strong class, we can simulate corresponding classes of proof systems by non-interactive protocols in the CRS model (i.e., by a two-message public-coin protocol).

The assumptions in this work (cf., Theorems 1.3 and 1.7) refer to a single function, rather than to a keyed collection of functions. This is also the case in previous works that constructed targeted hitting-set generators (targeted HSGs) and targeted pseudorandom generators (targeted PRGs), which are a single function rather than a keyed collection [CT23]. Indeed, we can use targeted HSGs to simulate corresponding classes of proof systems by non-interactive protocols in the *plain* model (i.e., by cs-$\mathcal{NTIME}$ verifiers).

To fairly compare the two objects, we need to compare either "keyed" targeted PRGs to standard CIHFs, or standard targeted HSGs to "keyless" CIHFs.[40] In the current section we take the latter route, but it will be clear from our proofs that the two "keyed" versions are also equivalent. Moreover, "keyed" versions of all our assumptions in this paper allow simulating the corresponding proof systems in the CRS model, as one would expect.

[40] It is well-known that when considering CIHFs against a class of relations $\mathcal{R} \supseteq \mathcal{P}/\text{poly}$, efficiently computable keyless CIHFs do not exist (see [CGH04]). However, as we explain below, we consider CIHFs h for relations whose complexity is lower than that of h itself. In fact, in Theorem 6.8 we provide evidence that in this setting, keyless CIHFs (of a relaxed type) do exist.

6.1 Targeted generators

Targeted PRGs were first defined by Goldreich [Gol11], who showed that they are sufficient and necessary for derandomization. Specifically, he considered targeted PRGs that are pseudorandom for uniform bounded-time algorithms on all inputs, and showed an equivalence between such objects (running in fixed polynomial time) and the assumption $prBPP = prP$.

Following [CT21; CT23], we consider a more general notion. First, we consider targeted HSGs rather than targeted PRGs. Secondly, we allow classes of uniform algorithms other than just bounded-time algorithms. And thirdly, we do not necessarily require pseudorandomness on all inputs, but also allow pseudorandomness only over certain distributions. That is:

Definition 6.1 (ϵ-avoiding)**.** *We say that a function $D\colon \{0,1\}^m \to \{0,1\}$ is an $(1-\epsilon)$-*dense avoider *for a set $S \subseteq \{0,1\}^m$ if*

$$\Pr_{r\in\{0,1\}^m}[D(r)=1] \geq 1-\epsilon \quad \wedge \quad \forall s \in S,\ D(s)=0\,.$$

Definition 6.2 (targeted HSGs)**.** *Let G be an algorithm that gets input $x \in \{0,1\}^n$ and outputs $p = p(n)$ strings of length $m = m(n)$, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions (where $\mathbf{x}_n$ is over $\{0,1\}^n$). We say that G is a* targeted hitting-set generator that is (ϵ,δ)-pseudorandom over $\mathbf{x}$ for a class $\mathcal{C}$ of uniform algorithms *(or (ϵ,δ)-*targeted HSG over $\mathbf{x}$ *for $\mathcal{C}$, in short) if for every Turing machine $M \in \mathcal{C}$, and every sufficiently large $n \in \mathbb{N}$,*

$$\Pr_{x\sim\mathbf{x}_n}[M(x,\cdot) \text{ is a } (1-\epsilon)\text{-dense avoider for } G(x)] \leq \delta(n)\,, \tag{6.1}$$

where $M(x,\cdot)$ is the function that gets input r and outputs $A(x,r)$.

6.2 List-correlation-intractable hash functions

Our goal now is to define a relaxed type of (keyless) correlation-intractable hash functions. Since we will be dealing with keyless CIHFs (which, by themselves, are not commonly studied), let us first define the natural keyless variant of CIHFs, before presenting its relaxation.

Definition 6.3 (sparse relations)**.** *We say that a relation $R_n \subseteq \{0,1\}^n \times \{0,1\}^m$ is ϵ-*sparse *if for every $x \in \{0,1\}^n$ we have that $\Pr_{r\in\{0,1\}^m}[(x,r) \in R_n] \leq \epsilon$. When $R = \{R_n \subseteq \{0,1\}^n \times \{0,1\}^m\}_{n\in\mathbb{N}}$ is a sequence of relations, we require that R_n is $\epsilon(n)$-sparse for every sufficiently large $n \in \mathbb{N}$.*

Definition 6.4 (recognizable relations)**.** *Let $\mathcal{C}$ be a class of uniform algorithms. We say that a relation $R = \{R_n \subseteq \{0,1\}^n \times \{0,1\}^m\}_{n\in\mathbb{N}}$ is* recognizable in $\mathcal{C}$ *if there is a Turing machine $M \in \mathcal{C}$ that gets input $(x,r) \in \{0,1\}^n \times \{0,1\}^m$ and accepts if and only if $(x,r) \in R_n$.*

When $\mathcal{C}$ is a class of bounded-time or bounded-space machines, we will measure the machine's complexity as a function of x, rather than of (x,r). This choice is useful for our equivalence result, but it also fits well with applications of CIHFs: Typically there is a "main input" x, or a security parameter λ, and complexity/security is measured with respect to x or to λ.

Definition 6.5 (keyless CIHFs)**.** *Let h be a function that maps n bits to $m = m(n)$ bits, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions (where $\mathbf{x}_n$ is over $\{0,1\}^n$). We say that h is δ-*correlation-intractable over $\mathbf{x}$ for ϵ-sparse relations recognizable in $\mathcal{C}$ *if for every ϵ-sparse relation $R = \{R_n \subseteq \{0,1\}^n \times \{0,1\}^m\}_{n\in\mathbb{N}}$ recognizable in $\mathcal{C}$, and every sufficiently large $n \in \mathbb{N}$,*

$$\Pr_{x\sim\mathbf{x}_n}[(x,h(x)) \in R_n] \leq \delta(n)\,. \tag{6.2}$$

Definition 6.5 refers to CIHFs that are secure over a fixed distribution $\mathbf{x}$. This also allows us to consider CIHFs that are secure over every $\mathbf{x}$ coming from a class $\mathcal{X}$ of distributions. For example, the standard notion of CIHFs considers $\mathcal{X}$ as the class of all efficiently samplable distributions.

Recall that Canneti, Goldreich, and Halevi [CGH04] showed that there is no polynomial-time computable keyless CIHF secure against all relations in $\mathcal{P}/\text{poly}$. However, their result does not preclude a keyless CIHF computable in time (say) T^2 that is correlation-intractable for relations recognizable in time T. (In fact, in Theorem 6.8 below we present strong evidence for the existence of (our relaxed variant of) keyless CIHFs.) Note that CIHFs whose complexity is larger than that of recognizing R, as is allowed in Definition 6.5, suffice for the key application of Fiat-Shamir.

The relaxed variant: List-CIHFs. We now present the relaxed variant of keyless CIHFs. In this relaxation, we allow h to output multiple strings $h(x)_1, ..., h(x)_p$, and only ask that for any sparse and efficiently recognizable relation R, *at least one* of the output strings $h(x)_s$ is such that $(x, h(x)_s)$ avoids R. This natural relaxation has not been considered in the literature before.

Definition 6.6 (list-CIHFs)**.** *Let h be a function that gets input $x \in \{0,1\}^n$ and outputs $p = p(n)$ strings of length $m = m(n)$, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions. We say that h is δ-list-correlation-intractable over $\mathbf{x}$ for ϵ-sparse relations recognizable in $\mathcal{C}$ if for every ϵ-sparse relation $R = \{R_n \subseteq \{0,1\}^n \times \{0,1\}^m\}_{n\in\mathbb{N}}$ recognizable in $\mathcal{C}$, and every sufficiently large $n \in \mathbb{N}$,*

$$\Pr_{x\sim\mathbf{x}_n}[\forall s \in [p(n)] \ : \ (x, h(x)_s) \in R_n] \leq \delta(n) \ . \tag{6.3}$$

We will consider efficiently computable list-CIHFs, in which case the list size $p = p(n)$ is also bounded (by the runtime of h). Note that a "keyed" version of Definition 6.6 can be obtained in the natural way: Allowing h_α to depend on a random key α, and requiring that for every R and sufficiently large $n \in \mathbb{N}$, with high probability over α Eq. (6.3) holds.

6.3 The equivalence and its implications

We now show that the two notions in Definitions 6.2 and 6.6, one notion from complexity theory and the other notion from cryptography (respectively), are precisely identical.

Theorem 6.7 (targeted HSGs vs list-CIHFs)**.** *Let $\mathcal{C}$ be a class of algorithms running in time T. A function G is a targeted HSG (ϵ, δ)-pseudorandom over $\mathbf{x}$ for $\mathcal{C}$* if and only if *G is δ-list-correlation-intractable over $\mathbf{x}$ for ϵ-sparse relations recognizable in $\mathcal{C}$.*

For simplicity, we stated Theorem 6.7 with respect to the class $\mathcal{C}$ of algorithms running in time T, but the proof applies essentially as-is to many other classes (e.g., to algorithms running in linear space, or to classes of oracle machines).

Proof of Theorem 6.7. We first prove that a targeted HSG implies a list-CIHF. Let G be a targeted HSG (ϵ, δ)-pseudorandom over $\mathbf{x}$ for $\mathcal{C}$, let $h = G$ and let $R = \{R_n\}$ be an ϵ-sparse relation recognizable in $\mathcal{C}$. We define a probabilistic algorithm A that gets input x and randomness r, and accepts iff $(x, r) \notin R$. Note that A runs in time $T(|x|)$. Also, for each x we have that $\Pr_r[A(x, r) = 1] \geq 1 - \epsilon$, because R is sparse. In particular, if $\{(x, h(x)_s)\}_{s\in[p(n)]} \subseteq R_n$, then $A(x, \cdot)$ is a $(1 - \epsilon)$-dense avoider for $G(x)$. Thus, if Eq. (6.3) is violated for R, then Eq. (6.1) is violated for A.

In the other direction, let h be list-correlation-intractable for ϵ-sparse relations recognizable in $\mathcal{C}$, let $G = h$, and let A be a Turing machine that gets input (x, r), runs in time $T(|x|)$, and outputs a bit. We say that an input x is valid if $\Pr_r[A(x, r) = 1] \geq 1 - \epsilon$. We define a relation R that includes all pairs (x, r) such that x is valid and $A(x, r) = 0$. By definition, we have that R is ϵ-sparse and recognizable in time T. Now, for every x such that $A(x, \cdot)$ is a $(1 - \epsilon)$-dense avoider for $G(x)$, we know that x is valid and that for all $s \in [p(n)]$ it holds that $A(x, h(x)_s) = 0$; in particular, $\{(x, h(x)_s)\}_{s \in [p(n)]} \subseteq R_n$. Thus, if Eq. (6.1) is violated for A, then Eq. (6.3) is violated for R. ■

List-CIHFs for weaker relations are equivalent to $prBPP = prP$. The list-CIHFs (equiv., targeted HSGs) that suffice for the applications in this work and in [CT23] need to avoid relations decidable by bounded-space machines, or by algorithms with access to a non-trivial oracle.[41] List-CIHFs that avoid weaker classes of relations turn out to be equivalent to standard derandomization: For example, list-CIHFs avoiding relations recognizable in fixed polynomial time that are secure on all inputs exist *if and only if* $prBPP = prP$.

In more detail, we say that a targeted HSG is ϵ-pseudorandom for $\mathcal{C}$ on almost all inputs if it is $(\epsilon, 0)$-pseudorandom for $\mathcal{C}$ over every possible ensemble $\mathbf{x}$. Similarly, we say that a list-CIHF is list-correlation-intractable for $\mathcal{C}$ on almost all inputs if it is 0-list-correlation-intractable for $\mathcal{C}$ over every possible ensemble $\mathbf{x}$. Then:

Theorem 6.8 (list-CIHFs vs derandomization)**.** *The following three statements are equivalent:*

1. $prBPP = prP$.
2. *For every time bound $T(n) \geq n$, there exists a targeted HSG computable in time* $\mathrm{poly}(T)$ *that is $(1/2, 0)$-pseudorandom for time T on almost all inputs.*
3. *For every time bound $T(n) \geq n$, there exists h computable in time* $\mathrm{poly}(T)$ *that is list-correlation-intractable for $(1/2)$-sparse relations recognizable in time T on almost all inputs.*

Proof. The equivalence of Items (2) and (3) follows from Theorem 6.7. The implication (2) $\Rightarrow$ (1) follows because Item (2) implies that $prRP = prP$, and using the well-known fact that $prBPP = prRP^{prRP}$ (see [Lau83; Sip83; BF99; GZ11]). The implication (1) $\Rightarrow$ (2) follows using Goldreich's [Gol11] search-to-decision reduction. ■

We stress again that list-CIHFs as in Theorem 6.8 are not known to suffice for securely instantiating Fiat-Shamir.

7 List-CIHFs (equiv., targeted HSGs) suffice for Fiat-Shamir

The purpose of this section is to argue that the relaxation of CIHFs to list-CIHFs (as defined in Section 6.2) suffices for the application of Fiat-Shamir in several important settings.

In particular, we show that list-CIHFs with sufficiently small list size suffice for all of our results. We first prove that list-CIHFs suffice for applying the Fiat-Shamir heuristic to the proof

[41] In [CT23, Theorem 7.8] the relation is decidable by a bounded-time algorithm with oracle to a doubly-efficient interactive proof. In this work, the relation is decidable in bounded space (cf., Theorem 1.3) or in bounded-time with oracle access to bounded non-deterministic time (cf., Theorem 1.8).

system of [GKR15] (see Section 7.1), and we then prove that list-CIHFs suffice for applying the Fiat-Shamir heuristic to zero-knowledge PCPs, obtaining NIZK for $\mathcal{NP}$ (see Section 7.2).

7.1 List-CIHFs suffice for Fiat-Shamir of GKR

We define a class of proof systems called batchable proof systems, and show that list-CIHFs suffice for Fiat-Shamir of such systems. We then show that the proof system of [GKR15] is batchable, and deduce that list-CIHFs suffice for applying Fiat-Shamir to it.

Since the definition of batchable proof systems is quite cumbersome, we first recall in Section 7.1.1 the definition of proof systems with round-by-round soundness (introduced in [CCH+19]), which are a superclass of batchable proof systems. Then, in Section 7.1.2 we explain how to refine this definition to obtain the definition of batchable proof systems. In Section 7.1.3 we prove that list-CIHFs suffice for Fiat-Shamir of batchable proof systems. Lastly, in Section 7.1.4 we prove that the proof system of [GKR15] is batchable.

Notation. Throughout this section we will refer to promise problems that represent proof systems. This notion will be formally defined below, but at a high level, a promise problem $\Pi = (\mathsf{Y}, \mathsf{N})$ can represent a proof system by including in Y partial transcripts that are "good" (e.g., likely to be accepted by the verifier when communicating with an honest prover) and including in N partial transcripts that are "bad" (e.g., likely to be rejected by the verifier, regardless of the prover).

7.1.1 Round-by-round soundness

For illustration, and to set up the stage towards defining batchable proof systems, let us recall the more relaxed notion of proof systems with round-by-round soundness, which was introduced by Canetti *et al.* [CCH+19].

Definition 7.1 (round-by-round soundness)**.** *Let $\Pi_0 = (\mathsf{Y}_0, \mathsf{N}_0)$ be a promise problem, and let $\Pi = (\mathsf{Y}, \mathsf{N})$ be another promise problem. We say that Π* represents a proof system with *R* rounds and round-by-round soundness for *Π_0 if it satisfies the following three conditions:*

1. **(First round.)** *For every $x \in \{0,1\}^*$ it holds that:*[42]

$$x \in \mathsf{Y}_0 \implies \exists \pi_1 : \Pr_{r_1}\left[\exists \pi_2 : \langle x, \pi_1, r_1, \pi_2 \rangle \in \mathsf{Y}\right] = 1 \,,$$
$$x \in \mathsf{N}_0 \implies \forall \pi_2 : \Pr_{r_1}\left[\forall \pi_2 : \langle x, \pi_1, r_1, \pi_2 \rangle \in \mathsf{N}\right] \geq 1 - 1/3R \,.$$

2. **(Generic round.)** *For every $i \in \{2, ..., R-1\}$:*

$$\langle x, \pi_1, r_1, ..., r_{i-1}, \pi_i \rangle \in \mathsf{Y} \implies \Pr_{r_i}\left[\exists \pi_{i+1} : \langle x, \pi_1, r_1, ..., r_i, \pi_{i+1} \rangle \in \mathsf{Y}\right] = 1$$
$$\langle x, \pi_1, r_1, ..., r_{i-1}, \pi_i \rangle \in \mathsf{N} \implies \Pr_{r_i}\left[\forall \pi_{i+1} : \langle x, \pi_1, r_1, ..., r_i, \pi_{i+1} \rangle \in \mathsf{N}\right] \geq 1 - 1/3R$$

[42] We separate the first round from subsequent rounds because the first round refers to the initial promise-problem Π_0, and because in the proof systems considered in this paper, the prover speaks first. We could also use an alternative definition that forces consistency between Π_0 and Π (i.e., $x \in \mathsf{Y}_0 \iff x \in \mathsf{Y}$ and $x \in \mathsf{N}_0 \iff x \in \mathsf{N}$) and allows the verifier to speak first.

3. **(Verifiability.)** *There is a ppt verifier V such that*

$$\langle x, \pi_1, r_1, ..., r_{R-1}, \pi_R \rangle \in \mathsf{Y} \Longrightarrow \Pr\left[V(x, \pi_1, r_1, ..., r_{R-1}, \pi_R) = 1\right] = 1$$
$$\langle x, \pi_1, r_1, ..., r_{R-1}, \pi_R \rangle \in \mathsf{N} \Longrightarrow \Pr\left[V(x, \pi_1, r_1, ..., r_{R-1}, \pi_R) = 1\right] \leq 1/3$$

Canetti *et al.* [CCH+19] proved that the sumcheck protocol of Lund *et al.* [LFK+92] and the proof system of Goldwasser, Kalai, and Rothblum [GKR15] have round-by-round soundness. Since the proof system of [GKR15] can be used to show that $\mathcal{IP} = \mathcal{PSPACE}$, it follows that $\mathcal{PSPACE}$ has a proof system with round-by-round soundness.

7.1.2 Batchable proof systems

We now define K-batchable proof systems, where $K = K(n)$ is a parameter. It is instructive to think of the value $K(n) = \text{poly}(n)$. To aid in parsing the definition, we first explain our intention behind the various notions and notations in the definition.

The notions and notations underlying the definition. Let $\Pi_0 = (\mathsf{Y}, \mathsf{N})$, and assume that there is $\Pi^{\mathsf{rbr}} = (\mathsf{Y}^{\mathsf{rbr}}, \mathsf{N}^{\mathsf{rbr}})$ representing a proof system with round-by-round soundness for Π_0. Loosely speaking, a promise-problem Π represents a K-batchable proof system if there is another proof system in which the following happens.

Both parties get a common input x, and the prover sends an initial message π_1. Then, in each round $i = 2, ..., R-1$, instead of sending a single challenge r_{i-1} and expecting to receive a good response π_i (i.e., as in the proof system represented by Π^{rbr}), the verifier sends a tuple of K challenges $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$, and the honest prover will respond with the K corresponding good responses $\pi_i^{(1)}, ..., \pi_i^{(K)}$. That is, instead of sending r_{i-1} and expecting π_i such that the transcript with (r_{i-1}, π_i) appended is in $\mathsf{Y}^{\mathsf{rbr}}$, the verifier sends $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$ and expects to receive $\pi_i^{(1)}, ..., \pi_i^{(K)}$ such that for every $k \in [K]$, the transcript with $(r_{i-1}^{(k)}, \pi_i^{(k)})$ appended is in $\mathsf{Y}^{\mathsf{rbr}}$. In addition, in every round the prover will send another message π_i' that aids batch-verification of the K messages simultaneously. The point is that in each subsequent round the verifier continues to send K challenges and receive $K+1$ responses (rather than branch to K^i challenges after i rounds). At the end of the interaction, the verifier tests the final transcript and outputs a decision.

Recall that in the proof system represented by Π^{rbr}, there are $\mathsf{Y}^{\mathsf{rbr}}$-transcripts, which remain in $\mathsf{Y}^{\mathsf{rbr}}$ with high probability over the verifier's challenge (given an appropriate response from the honest prover) and $\mathsf{N}^{\mathsf{rbr}}$-transcripts, which remain in $\mathsf{N}^{\mathsf{rbr}}$ with high probability (for any response by any prover). For K-batchable proof systems, we will define analogous notions of good transcripts and of doomed transcripts, referring to partial transcripts with K challenges and $K+1$ responses in each round, and in a similar way, require that (with high probability) good transcripts remain good, and doomed transcripts remain doomed.

To do so, we think of $\Pi = (\mathsf{Y}, \mathsf{N})$ as representing a mental experiment of what would happen if the verifier would have sent a *single* challenge and received a *single* response in the latest round of partial interaction. Specifically, fix a partial transcript with K challenges and $K+1$ responses in each round, and consider the mental experiment of conducting *next* round with a single challenge and response. We define good transcripts as ones where, with high probability over the verifier's challenge in the next round, the transcript will be in Y (given an appropriate response from the

honest prover); and we define doomed transcripts as ones where, with high probability over the verifier's challenge, the transcript will be in N (for any response by any prover).

When analyzing a round in actual the proof system, with K challenges and $K+1$ responses, we think of each challenge-response pair $(r_{i-1}^{(k)}, \pi_i^{(k)})$ (for $k \in [K]$) as yielding a separate partial transcript $\pi_{\leq i}^{(k)}$ for the mental experiment of Π, and require the following: If all K partial transcripts $\pi_{\leq i}^{(1)}, ..., \pi_{\leq i}^{(K)}$ are in Y, then the actual transcript $\pi_{\leq i}$ (which is obtained when the verifier sends all K challenges and receives $K+1$ responses) is good; and if at least one of the K partial transcripts $\pi_{\leq i-1}^{(k)}$ is in N, then the actual transcript $\pi_{\leq i}$ is doomed. This ensures that, with high probability, good transcripts remain good, and doomed transcripts remain doomed.

The full definition. For convenience, given a partial transcript $\pi_{\leq i-1} = \langle x, \pi_1, ... \rangle$ and strings r_{i-1} and π_i, we denote by $\langle \pi_{\leq i-1}, r_{i-1}, \pi_i \rangle$ the partial transcript that is obtained by appending r_{i-1} and π_i at the end of $\pi_{\leq i-1}$.

Definition 7.2 (batchability)**.** *Let $\Pi_0 = (\mathsf{Y}_0, \mathsf{N}_0)$ be a promise problem, and let $\Pi = (\mathsf{Y}, \mathsf{N})$ be another promise problem. We say that Π represents a K-batchable proof system with R rounds for Π_0 if the following holds.*

Let $\pi_{\leq i} = \left\langle x, \pi_1, (r_1^{(1)}, ..., r_1^{(K)}), (\pi_2^{(1)}, ..., \pi_2^{(K)}, \pi_2'), ..., (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i') \right\rangle$ be a partial transcript. (For $i = 1$, we consider $\pi_{\leq 1} = \langle x, \pi_1 \rangle$.) We say that

$$\pi_{\leq i} \text{ is good if } \Pr_{r_i}[\exists \pi_{i+1} : \langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{Y}] = 1$$
$$\pi_{\leq i} \text{ is doomed if } \Pr_{r_i}[\exists \pi_{i+1} : \langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{N}] \geq 1 - 1/3R \,.$$

Then, we require that:

1. **(First round.)** *(Intuitively, in the first round, the parties hold common input x and the prover sends a message π_1 to the verifier.) We require that for every $x \in \{0,1\}^*$,*

$$x \in \mathsf{Y}_0 \implies \exists \pi_1 : \langle x, \pi_1 \rangle \text{ is good} \,,$$
$$x \in \mathsf{N}_0 \implies \forall \pi_1 : \langle x, \pi_1 \rangle \text{ is doomed} \,.$$

2. **(Generic round.)** *Let $i \in \{2, ..., R-1\}$, and fix a partial transcript $\pi_{\leq i-1} = \left\langle x, \pi_1, (r_1^{(1)}, ..., r_1^{(K)}), (\pi_2^{(1)}, ..., \pi_2^{(K)}, \pi_2'), ..., (\pi_{i-1}^{(1)}, ..., \pi_{i-1}^{(K)}, \pi_{i-1}') \right\rangle$.*[43]

(Intuitively: The verifier sends K challenges $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$, and the prover answers with K responses $\pi_i^{(1)}, ..., \pi_i^{(K)}$ along with an additional response π_i'. The transcript at the end of this round is $\pi_{\leq i} = \left\langle \pi_{\leq i-1}, \left(r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}\right), \left(\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i'\right) \right\rangle$.)

We require that for every K challenges $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$ and K responses $\pi_i^{(1)}, ..., \pi_i^{(K)}$,

$$\forall k \in [K], \left\langle \pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)} \right\rangle \in \mathsf{Y} \implies \exists \pi_i' : \pi_{\leq i} \text{ is good} \,,$$
$$\exists k \in [K], \left\langle \pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)} \right\rangle \in \mathsf{N} \implies \forall \pi_i' : \pi_{\leq i} \text{ is doomed} \,.$$

[43] In case $i = 2$, we will have $\pi_{\leq 1} = \langle x, \pi_1 \rangle$.

3. **(Final round.)** *Consider the last round $i = R$, and let $\pi_{\leq R-1}$ be a partial transcript going into this round.*

   (Intuitively: The verifier sends K challenges $r_{R-1}^{(1)}, ..., r_{R-1}^{(K)}$, the prover answers with K responses $\pi_R^{(1)}, ..., \pi_R^{(K)}$ along with an additional response π'_R, and the transcript at the end of the interaction is $\pi_{\leq R} = \left\langle \pi_{\leq R-1}, \left(r_{R-1}^{(1)}, ..., r_{R-1}^{(K)}\right), \left(\pi_R^{(1)}, ..., \pi_R^{(K)}, \pi'_R\right)\right\rangle$.)

   Then, there is a probabilistic verification procedure V running in time $\mathrm{poly}(n)$ such that for every K challenges and K responses,

$$\forall k \in [K], \left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{Y} \implies \exists \pi'_R : \Pr[V(\pi_{\leq R}) = 1] = 1 ,$$
$$\exists k \in [K], \left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{N} \implies \forall \pi'_R : \Pr[V(\pi_{\leq R}) = 1] \leq 1/3 .$$

The randomness complexity *of V is the maximum, over all $i \in [R]$ and partial transcripts $\pi_{\leq i}$ up to round i, of the length of r_i. We assume, for simplicity, that this value also upper-bounds the number of random coins that V uses in the final verification procedure. The* next-prover-message complexity *is the complexity of deciding, given a partial transcript $\pi_{\leq i} = \langle x, \pi_1, \hat{r}_1, ..., \hat{\pi}_i, r_i\rangle$, whether or not there exists π_{i+1} such that $\langle \pi_{\leq i}, \pi_{i+1}\rangle \notin \mathsf{N}$.*

Remark 7.3. *Batchable proof systems are a special case of proof systems with round-by-round soundness. To see this, observe that if a proof system is K-batchable, then it is K'-batchable for any $K' < K$ (i.e., the verifier can simply repeat some challenges to get precisely K challenges); and note that when $K = 1$, Definition 7.2 is essentially identical to Definition 7.1.*[44] *Thus, if a proof system is K-batchable, in particular it has round-by-round soundness.*

7.1.3 List-CIHFs suffice for Fiat-Shamir of batchable proof systems

We now present a general result asserting that targeted hitting-set generators suffice for derandomization of batchable proof systems. Since targeted HSGs are equivalent to list-CIHFs (see Theorem 6.7), this can be equivalently thought of as showing that list-CIHFs suffice for applying Fiat-Shamir to batchable proof systems. Indeed, we arbitrarily chose to use the former terminology in the statement and proof rather than the latter.

Theorem 7.4 (targeted HSGs suffice for derandomization of batchable proof systems)**.** *Let K, T_V, T_G, T_P be time bounds such that $K, T_V, T_G \leq 2^{o(n)}$.*

- *Let $\Pi = (\mathsf{Y}, \mathsf{N})$ be a promise problem that has a K-batchable proof system such that the verifier runs in time T_V and has randomness complexity $r = r(n)$, the next-prover-message space complexity is linear, the honest prover runs in time T_P, the number of rounds is $R = R(n) \leq T_P(n)$, and the total communication complexity is $\mathsf{cc} = \mathsf{cc}(n)$. Also assume that the verification in the final round is deterministic.*[45]

[44] The only difference between the definitions when $K = 1$ is that batchable proof systems allow the prover to send an auxiliary message π'_i at each round; however, this auxiliary message is redundant when $K = 1$.

[45] Alternatively, we may assume that the space complexity of verifying the final transcript is linear (i.e., space complexity $O(\mathsf{cc}(n))$). The point is that if the last verification step is probabilistic, we need the targeted HSG to be pseudorandom for it.

- *For $T'_P = O(K \cdot T_P)$ and $N = N(n) \geq \mathsf{cc}(n)$, assume that there is a targeted HSG G that is $(1/2, T'_P(n)^{-\omega(1)})$-pseudorandom for space $O(\mathsf{cc}(n))$ over all distributions samplable in time $\mathrm{poly}(T'_P(n))$ such that on inputs of length N, the targeted HSG runs in time $T_G(N)$ and prints $K(n)$ strings of length $r(n)$. (We think of the targeted HSG as instantiated on inputs z of length $N(n) \geq n$, so all occurrences of n in the foregoing statement mean $N^{-1}(|z|)$.)*

Then, $\Pi \in \mathsf{cs}\text{-}\mathcal{NTIME}[\mathrm{poly}(T_V, R \cdot T_G(N)), T'_P]$.

A useful setting of parameters to keep in mind is the following. For $K(n) = \mathrm{poly}(n)$, the K-batchable proof system has $R(n) = O(n)$ rounds, with a verifier running in polynomial time $T_V(n) = \mathrm{poly}(n)$ and sending $r(n) = O(n)$ bits in each challenge. The targeted HSG gets input of length $N = \mathrm{poly}(n)$, runs in polynomial time, and outputs $K(n) = n^{o(1)}$ strings of length r.

The proof system of [GKR15] can be used to decide TQBF with such parameters (see Section 7.1.4), and a targeted HSG with the parameters above was constructed in [CT21; CT23] from hardness over all efficiently samplable distributions (see Corollary 7.6).

Proof of Theorem 7.4. We define a deterministic verifier V that interacts with the prover in rounds. The existence of such a verifier implies the existence of an $\mathcal{NP}$-verifier for the same problem, since the honest prover can send all messages in advance to the $\mathcal{NP}$-verifier.

The deterministic verifier. At each round $i \in [R-1]$, let $\pi_{\leq i} = \left\langle x, \pi_1, (r_1^{(1)}, ..., r_1^{(K)}), ..., (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi'_i) \right\rangle$ be the transcript going into round i. The verifier pads $\pi_{\leq i}$ by zeroes to be of N bits, and computes G on the padded string, to obtain the pseudorandom strings $r_i^{(1)}, ..., r_i^{(K)}$. In the last round, the verifier accepts if and only if the original verifier accepts the transcript.

The only overhead of V on top of the original verifier is in computing G in each of the R rounds. Thus, the verifier V runs in time $T_V \cdot R \cdot T_G(N) \leq \mathrm{poly}(T_V, T_G, R)$, and the $\mathcal{NP}$-verifier runs in the same time.

Completeness. Note that V chooses a pseudorandom subset of K challenges in each round, and uses a final verification procedure identical to that of the original verifier. Thus, completeness is trivially maintained, using essentially the same honest prover as in the original proof system.

In more detail, in each round in the new protocol, and given each of the K challenges, the honest prover needs to compute the same response that the honest prover in the original system would have sent. Thus, the honest prover's time complexity is at most $T'_P = O(K \cdot T_P)$.

Soundness. To prove soundness, consider any efficient adversary $\tilde{P}$ as a distribution over deterministic efficient strategies, and denote by $P \sim \tilde{P}$ a choice of deterministic efficient strategy. We will prove that the following holds:

Claim 7.4.1 (round-by-round soundness is maintained)**.** *For any fixed $i \in [R-1]$ and fixed $P \sim \tilde{P}$, let $\pi_{\leq i} = \pi^P_{\leq i} = \left\langle x, \pi_1, (r_1^{(1)}, ..., r_1^{(K)}), (\pi_2^{(1)}, ..., \pi_2^{(K)}, \pi'_2), ..., (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi'_i) \right\rangle$ be the transcript of interaction between V and P up to round i, and let $r_i^{(1)}, ..., r_i^{(K)}$ be the challenges that V chooses. We say that a* verification mistake *occurred if $\pi_{\leq i}$ is doomed and*

$$\forall k \in [K] \; \exists \pi_{i+1}^{(k)} \; : \; \left\langle x, \pi_1, ..., (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi'_i), r_i^{(k)}, \pi_{i+1}^{(k)} \right\rangle \notin \mathsf{N} \,. \tag{7.1}$$

Then, the probability over $P \sim \tilde{P}$ that a verification mistake occurred is $\mathsf{negl}(T'_P(n))$.

Proof. For $i \in [R-1]$, we say that $\pi_{\leq i}$ is bad if $\pi_{\leq i}$ is doomed and Eq. (7.1) holds (i.e., if a verification mistake occurred). Assume towards a contradiction that there exists a probabilistic T'_P-time adversary $\tilde{P}$ such that for infinitely many $n \in \mathbb{N}$, with probability at least $\mathrm{poly}(1/T'_P)$ over $P \sim \tilde{P}$, there exists $i \in [R]$ such that $\pi_{\leq i}$ is bad.

Let $\mathbf{z}_N$ be the distribution over N-bit inputs obtained by randomly choosing $P \sim \tilde{P}$ and $i \in [R]$, simulating the interaction between V and P up to round i to obtain $\pi_{\leq i}$, and padding $\pi_{\leq i}$ by zeroes to obtain an N-bit string τ_i. Note that $\mathbf{z} = \{\mathbf{z}_N\}_N$ is samplable in time $\mathrm{poly}(T'_P(n))$.

For every τ_i in the support of $\mathbf{z}$, let D_{τ_i} be the function that gets input r_i and outputs "yes" if and only if for all π_{i+1} it holds that $(\pi_{\leq i}, r_i, \pi_{i+1}) \in \mathsf{N}$. Note that if $\pi_{\leq i}$ is bad, then $\Pr_{r_i}[D_{\tau_i}(r_i) = 1] \geq 1 - 1/3R$, but $\Pr_{k \in [K]}\left[D_{\tau_i}(r_i^{(k)}) = 1\right] = 0$. Hence, whenever $\pi_{\leq i}$ is bad, D_{τ_i} is a $(1/2)$-avoider for $G_f(\tau_i)$. Also, since the next-prover-message space complexity in the proof system is linear, the function $\mathsf{Next}(\tau_i, r_i) = D_{\tau_i}(r_i)$ is also computable in space $O(\mathsf{cc}(n))$.

Since the probability over $z_N \sim \mathbf{z}_N$ that τ_i corresponds to a bad $\pi_{\leq i}$ is at least $\mathrm{poly}(1/T'_P(n)) \cdot (1/R) = \mathrm{poly}(1/T'_P(n))$, the probability over $z_N \sim \mathbf{z}_N$ that Next is a $(1/2)$-avoider for $G(z_N)$ is at least $\mathrm{poly}(1/T'_P(n))$, a contradiction. □

By a union-bound over the R rounds, with probability at least $1 - \mathsf{negl}(T'_P)$ that there was no verification mistake in any of the rounds. In particular, if $x \in \mathsf{N}$ then with probability at least $1 - \mathsf{negl}(T'_P)$ the final transcript $\pi_{\leq R}$ is such that the verifier rejects. ■

7.1.4 The GKR protocol is batchable

We now argue that the proof system of Goldwasser, Kalai, and Rothblum [GKR15] is batchable, in the sense of Definition 7.2. Since our goal is to demonstrate that list-CIHFs suffice to instantiate the Fiat-Shamir heuristic with this proof system (i.e., apply Theorem 7.4 to this proof system), we will also state the various complexity parameters of this proof system (e.g., the next-prover-complexity, the honest prover complexity, and so on).

Proposition 7.5 (the GKR protocol is batchable)**.** *There exist two universal constants c, c' such that the following holds. Let $\{C_n\}_{n \in \mathbb{N}}$ be a logspace-uniform family of circuits of size $T(n)$ and depth $d(n)$, and let $\gamma \colon \mathbb{N} \to (0,1)$ be a logspace-computable function such that $T(n)^{\gamma(n)} \geq \log(T(n))$ and $d(n) \leq T(n)^{c \cdot \gamma(n)}$.*

Let $\Pi_0 = (\mathsf{Y}_0, \mathsf{N}_0)$ where $(x_0, b) \in \mathsf{Y}_0$ if $C_{|x_0|}(x_0) = b$ and $(x_0, b) \in \mathsf{N}_0$ if $C_{|x_0|}(x_0) = \neg b$. Then, for any $K(n) \leq T(n)^{c \cdot \gamma}/d(n)$, there is a promise problem Π representing a proof system for Π_0 with the following properties:

1. *The proof system is K-batchable.*
2. *There are $R = O((d/\gamma) \cdot \log(T))$ rounds. The first message by the prover is of length $T^\gamma \cdot \mathrm{polylog}(T)$. In each subsequent round, each of the K messages of the verifier is of length $O(\gamma \cdot \log(T))$, and the prover sends $K+1$ messages, each of length at most $K \cdot T^\gamma \cdot \mathrm{polylog}(T)$.*
3. *The final verification procedure runs in time $K^3 \cdot T^{c' \cdot \gamma} \cdot n \cdot d \cdot \mathrm{polylog}(T)$.*

4. *The honest prover's strategy in each round is computable in time* $T^{(1+\gamma)\cdot c'} \cdot K^3 \cdot d \cdot \mathrm{polylog}(T)$. *In addition, if the mapping of* $(x, g) \in \{0,1\}^n \times [T(n)]$ *to the value of the* g^{th} *gate in* $C_n(x)$ *is computable in space* S*, then the next-prover-message space complexity is* $O((d/\gamma) \cdot \log(T \cdot K)^2 + S)$.

Let us explain how the useful setting of parameters, which was suggested after the statement of Theorem 7.4, can be obtained from Proposition 7.5. Recall that the $\mathcal{PSPACE}$-complete problem TQBF has logspace-uniform circuits of size $T(n) = 2^{O(n)}$ and depth $d(n) = \mathrm{poly}(n)$, in which the value of each gate in circuit (on a given input x) can be computed in linear space. Let $K(n) = n^c$ for some constant c, and let $\gamma(n) = O(\mathrm{loglog}(T)/\log(T))$ (such that $T^{c\cdot\gamma} = \mathrm{poly}(d)$). The proof system in Proposition 7.5 has $R(n) = \mathrm{poly}(n)$ rounds, with a verifier running in polynomial time and sending $r(n) = O(\mathrm{loglog}(T)) = O(\log n)$ bits in each challenge, and with a next-prover-message space complexity $\mathrm{poly}(n)$. For an appropriately padded version of TQBF (which is still $\mathcal{PSPACE}$-complete), we get a proof system with linearly many rounds and with linear next-prover-message space complexity.

By combining the logspace-uniform circuits for the padded version of TQBF, the batchable proof system of Proposition 7.5, and the cs-$\mathcal{NTIME}$ simulation from Theorem 7.4, we obtain the following corollary:

Corollary 7.6. *Assume that there is a targeted HSG computable in polynomial time that is* $(1/2, 2^{-O(n)})$*-pseudorandom for linear space over all distributions samplable in time* $2^{O(n)}$*, such that on inputs of length* N *the targeted HSG outputs at most* N^ϵ *strings of length* $\Theta(\log N)$*, where* $\epsilon > 0$ *is a sufficiently small constant. Then, there is a* $\mathcal{PSPACE}$*-complete problem decidable in* cs-$\mathcal{NTIME}[\mathrm{poly}(n), 2^{O(n)}]$.

Indeed, the running time $2^{O(n)}$ of the adversary in the assumption and of the prover in the conclusion can be improved to 2^{n^ϵ} (for any constant $\epsilon > 0$), using sufficient padding.

7.1.4.1 Polynomial decomposition of a circuit

For the proof of Proposition 7.5, we will need the following result, which is a variant of a result from the previous work [CRT22, Proposition 4.2], following [GKR15; CT21].

Proposition 7.7 (polynomial decomposition of a circuit)**.** *There exist two universal constants* $c_0, c_0' \in \mathbb{N}$ *such that the following holds. Let* $\{C_n\}_{n\in\mathbb{N}}$ *be a logspace-uniform family of circuits of size* $T(n)$ *and depth* $d(n)$*, and let* $\gamma\colon \mathbb{N} \to (0,1)$ *be a logspace-computable function such that* $T(n)^{\gamma(n)} \geq \log(T(n))$*. Then, for* $T'(n) = O(T(n)^{c_0})$ *and* $d'(n) = O(d(n) \cdot \log(T(n)))$ *there is a circuit family* $\{C_n'\}_{n\in\mathbb{N}}$ *of size* T' *and depth* d' *computing the same function as* $\{C_n\}$*, such that for every* $x \in \{0,1\}^n$ *there exists a sequence of polynomials* $\{\hat{\alpha}_i\colon \mathbb{F}^m \to \mathbb{F}\}_{i\in\{0,\ldots,d'\cdot m\}}$ *satisfying the following:*

1. **(Arithmetic setting.)** *The polynomials are defined over* $\mathbb{F}_p$*, where* p *is the smallest prime in the interval* $[T^{\gamma\cdot c_0}, 2T^{\gamma\cdot c_0}]$*. Let* $H = [h] \subseteq \mathbb{F}$*, where* h *is the smallest power of two of magnitude at least* $T^{\gamma/6}$*, and let* m *be the minimal integer multiple of* 3 *such that* $h^{m/3} \geq 2T^{c_0}$*. The total degree of each* $\hat{\alpha}_i$ *is at most* $\Delta = h \cdot \mathrm{polylog}(T)$.

2. **(Faithful representation.)** *For every* $i \in [d'(n)]$ *and* $\vec{w} \in H^m$ *representing a gate in the* i^{th} *layer of* C_n' *it holds that* $\hat{\alpha}_{(d'-i)\cdot m}(\vec{w})$ *is the value of the corresponding gate in* $C_n'(x)$.

3. **(Base layer.)** *There is an algorithm* BASE *that gets input* $x \in \{0,1\}^n$ *and* $\vec{w} \in \mathbb{F}^m$*, runs in time* $n \cdot h^{c_0'}$*, and outputs* $\hat{\alpha}_{d'\cdot m}(\vec{w})$.

4. **(Upwards self-reducibility.)** *There is an algorithm* USR *that gets input* 1^n *and* $i \in \{0,\ldots,d' \cdot m - 1\}$ *and* $\vec{w} \in \mathbb{F}^m$*, runs in time* $h^{c_0'}$ *while making* h *non-adaptive queries to* $\hat{\alpha}_{i+1}$*, and outputs* $\hat{\alpha}_i(\vec{w})$*. There is another implementation of* USR *that uses space* $O(\log T)$ *(instead of time* $h^{c_0'}$*).*

Furthermore, the mapping $(i,\vec{w}) \mapsto \hat{\alpha}_i(\vec{w})$ *is computable in time* $T(n)^{c_0'}$*; and if the mapping of* $(x,g) \in \{0,1\}^n \times [T(n)]$ *to the value of the* g^{th} *gate in* $C_n(x)$ *is computable in space* S*, then the mapping* $(i,\vec{w}) \mapsto \hat{\alpha}_i(\vec{w})$ *can be computed in space* $O(S + \log(T))$

Proposition 7.7 was essentially already proved in [CRT22, Proposition 4.2]. However, since we use different notation, and since some of the properties were demonstrated in their proof but not explicitly stated in their result statement, we now explain how to obtain Proposition 7.7.

Proof sketch for Proposition 7.7. Our starting point is [CRT22, Proposition 4.2], which uses an identical arithmetic setting except that they use the parameter value $m' = m/3$ (i.e., they denote by m the value that we denote $m/3$). For $d' = O(d \cdot \log T)$, they define polynomials

$$\left\{\hat{\beta}_{i,j} \colon \mathbb{F}^{m'+j} \to \mathbb{F}\right\}_{i\in[d'],j\in[2m']} ,$$
$$\left\{\hat{\beta}_{i} \colon \mathbb{F}^{m'} \to \mathbb{F}\right\}_{i\in\{0,\ldots,d'\}} ,$$

and they also define $\hat{\beta}_i \equiv \hat{\beta}_{i,0}$ for $i \geq 1$. (They denote the polynomials as $\hat{\alpha}_i$ and $\hat{\alpha}_{i,j}$, but since we will use a different indexing, we denote their polynomials as $\hat{\beta}_i$ and $\hat{\beta}_{i,j}$.)

By adding dummy variables, we can think of all polynomials as mapping $\mathbb{F}^m \to \mathbb{F}$. We also reindex the polynomials in the following order:

$$\underbrace{\hat{\beta}_{d',0}}_{\hat{\alpha}_0}, \underbrace{\hat{\beta}_{d',1}}_{\hat{\alpha}_1}, \ldots, \underbrace{\hat{\beta}_{d',2m'}}_{\hat{\alpha}_{2m'}}, \hat{\beta}_{d'-1,0}, \ldots, \hat{\beta}_{d'-1,2m'}, \ldots, \hat{\beta}_{1,0}, \ldots, \hat{\beta}_{1,2m'}, \underbrace{\hat{\beta}_0}_{\hat{\alpha}_{(2m'+1)\cdot d'}} ,$$

which yields a sequence of $(2m'+1)\cdot d' < m \cdot d' + 1$ polynomials. We add dummy polynomials to make the sequence of length exactly $m \cdot d' + 1$. The faithful representation follows immediately from the faithful representation in [CRT22] and our re-indexing.

In [CRT22] they prove that for every $i \in [d']$ and $j \in [2m']$, computing $\hat{\beta}_{i,j-1}$ reduces to computing $\hat{\beta}_{i,j}$; and that for every $i \in [d']$, computing $\hat{\beta}_{i,2m'}$ reduces to computing $\hat{\beta}_{i-1} = \hat{\beta}_{i-1,0}$. Their reduction is stated as a logspace-uniform oracle circuit of size $h^{c_0'}$ and bounded depth, and the construction readily gives a non-adaptive oracle machine USR with running time $h^{c_0'}$. Similarly, they show a logspace-uniform circuit of size $\max\{n,h\} \cdot h^{c_0'}$ (and bounded depth) computing $\hat{\beta}_0 = \hat{\alpha}_{m\cdot d+1}$, and their construction readily gives an algorithm BASE with running time $\max\{n,h\} \cdot h^{c_0'}$.

To see that the polynomials can be computed in time $\mathrm{poly}(T)$, note that in such time we can evaluate the entire circuit C_n' at input x, and then compute the polynomial extension of each layer that is defined in [CRT22, Definition 4.1 and Claim 4.2.2]. The fact that USR can be computed in space $O(\log T)$ is proved in [CRT22, Claim 4.2.2].

Finally, for the space complexity, in [CRT22, Proof of Claim 4.2.1.1] it is shown that the mapping of $(x,i) \in \{0,1\}^n \times \{0,1\}^{T'}$ to the value of the i^{th} gate in $C_n'(x)$ is computable in space $O(S + \log(T))$.[46] And from [CRT22, Claim 4.2.2] it follows that computing $\hat{\alpha}_i$ reduces in space $O(\log T)$ to computing the foregoing mapping. ■

[46] The original statement in [CRT22] is only for the special case $S = O(\log T)$, but the proof already shows the general case.

Given input $x = (x_0, b)$ where $|x_0| = n$, we consider the polynomial decomposition of C_n from Proposition 7.7, with the parameters $c_0, c_0', d', h, |\mathbb{F}|, \Delta, m$ specified in the proposition's statement. Observe that $m = O(1/\gamma)$, and denote $R = d' \cdot m - 1 = O(d \cdot \log(T)/\gamma)$.

7.1.4.2 The promise problem: Definition

Our first goal is to define the promise problem. Towards doing so, we start by describing the format of messages in the promise problem, and defining some preliminary notions and objects. We urge the reader to remind themselves of the notation in Definition 7.2.

The format of messages in the proof system. The interaction in the proof system is conducted as follows:

1. **The first round.** On common input x, the prover sends a univariate π_1 of degree $\Delta \cdot h$, yielding a transcript $\pi_{\leq 1} = \langle x, \pi_1 \rangle$.
2. **A generic round $i \in \{2, ..., R\}$.** Let $\pi_{\leq i-1}$ be the transcript going into round i. The verifier sends K elements $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)} \in \mathbb{F}$, and receives K univariates $\pi_i^{(1)}, ..., \pi_i^{(K)}$ of degree $\Delta \cdot h$ along with an additional univariate π_i' of degree $\Delta \cdot K \cdot h$. This yields a transcript $\pi_{\leq i} = \left\langle \pi_{\leq i-1}, \left(r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}\right), \left(\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i'\right) \right\rangle$.
3. **Partial transcripts ending with a single challenge and response.** As a mental experiment, we also consider partial transcripts where in round $i \in \{2, ..., R\}$ the verifier sends a *single* element r_{i-1} and the prover answers by a *single* univarite π_i of degree $\Delta \cdot h$. (This will be done to define $\Pi = (\mathsf{Y}, \mathsf{N})$ as in Definition 7.2.)

To simplify notation, we will frequently denote tuples of challenges and responses by $\hat{r}_{i-1} = \left(r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}\right)$ and $\hat{\pi}_i = \left(\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i'\right)$, respectively. For example, we denote a complete transcript by

$$\pi_{\leq R} = \langle x, \pi_1, \hat{r}_1, \hat{\pi}_2, ..., \hat{r}_{R-1}, \hat{\pi}_R \rangle \ ,$$

and we denote partial transcripts ending with a single challenge and response by

$$\pi_{\leq i} = \langle x, \pi_1, \hat{r}_1, \hat{\pi}_2, ..., \hat{\pi}_{i-1}, r_{i-1}, \pi_i \rangle \ .$$

The notation $\pi_{\leq i}$ does not indicate whether the partial transcript ended with a single challenge and response or with K challenges and $K+1$ responses, but our intention will be clear from context whenever we use the notation.

Intuition. To motivate the definitions that come next, let us give intuition to how the proof system works; this part may be safely skipped.

The goal of the verifier is to check that $\hat{\alpha}_0(\vec{w}) = 1$, where $\vec{w} = 0^m$ represents the index of the output gate of $C_n'(x)$. The verifier reduces this task to testing the values of $\hat{\alpha}_1$ at h points: Specifically, it runs USR, which queries $\hat{\alpha}_1$ at points $(q_{0,\vec{w},a})_{a \in [h]} \subseteq \mathbb{F}^m$. The verifier sends these

points to the prover,[47] and both the verifier and the prover interpolate a degree-h curve $\Gamma_1 : \mathbb{F} \to \mathbb{F}^m$ that passes through $\mathsf{Next}(0, \vec{w})$. The prover sends a univariate $\pi_1 : \mathbb{F} \to \mathbb{F}$ that is supposed to be the correct polynomial, defined as $p_1(r) = \hat{\alpha}_1(\Gamma_1(r))$, and the verifier checks that when the queries of USR are answered according to π_1, it indeed outputs 1.

In the next round, the verifier chooses random $r_1^{(1)}, ..., r_1^{(K)} \in \mathbb{F}$, and wants to verify that for all $k \in [K]$ it holds that $p_1(r_1^{(k)}) = p_1(\Gamma_1(r_1^{(k)})) = v_1^{(k)}$, where $v_1^{(k)} = \pi_1(r_1^{(k)})$. For each $k \in [K]$, the prover and verifier act similarly to the first round: They interpolate $\Gamma_2^{(k)}$ that passes through the h points that USR queries on input $(1, \Gamma_1(r_1^{(k)}))$ (i.e., through $(q_{1,\Gamma_1(r_1^{(k)}),a})_{a\in[h]}$), and the prover sends $\pi_2^{(k)}$ that is supposed to be $p_2^{(k)}(r) = \hat{\alpha}_2(\Gamma_2^{r_1^{(k)}}(r))$; the verifier checks that USR indeed answers with $v_1^{(k)}$.

To batch-verify all the $\pi_i^{(k)}$'s together, the prover and verifier also interpolate a polynomial Γ_2 that passes through all the $K \cdot h$ points (i.e., through $(q_{1,\Gamma_1(r_1^{(k)}),a})_{k\in[K],a\in[h]}$). The prover sends π_2' that is supposed to be $p_2(r) = \hat{\alpha}_2(\Gamma_2(r))$, and the verifier checkes that π_2' is consistent with $\pi_2^{(k)}$ on all the $K \cdot h$ points. At the next round, the verifier will choose random $r_2^{(1)}, ..., r_2^{(K)} \in \mathbb{F}$, and wants to verify that $\pi_2'(r_2^{(k)}) = p_2(\Gamma_2(r_2^{(k)}))$ for all $k \in [K]$.

Since K and h are small compared to $\mathbb{F}$, all polynomials are of low-degree. Thus, if $\pi_i \neq p_i$ for some i, then with high probability over r_i will disagree on r_i. The prover will then have to lie about the value of points in the next round, so that when USR queries these points, it outputs the (incorrect) value $\pi_i(r_i)$. At the next round, the final task is checking the univariate polynomial $\hat{\alpha}_{d' \cdot m} \circ \Gamma_{d' \cdot m}$, which can be done efficiently by the verifier (using the algorithm BASE). Details follow.

Preliminary definitions. Towards defining the promise problem Π, we will need to define a number of objects that will be part of the proof.

- **The first elements of $\mathbb{F}$.** For $(k, a, b) \in [K] \times [h] \times [\Delta + 1]$, let $\sigma_{k,a,b}$ be the $(k, a, b)^{th}$ element in $\mathbb{F}$ by lexicographic ordering of the tuples (k, a, b). For $(k, a) \in [K] \times [h]$, let $\sigma_{k,a} = \sigma_{k,a,1}$.
- **The USR queries.** For every $i \in \{0, ..., R-1\}$ and $\vec{w} \in \mathbb{F}^m$, let $(q_{i,\vec{w},a})_{a\in[h]}$ be the list of h queries issued by USR on input $(i, \vec{w})$.
- **The USR curves.** Fix $i \in \{2, ..., R\}$, let $\pi_{\leq i-1} = \langle x, \pi_1, ..., \hat{\pi}_{i-1} \rangle$ be a partial transcript, and let $\hat{r}_{i-1} = r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$.

  We define $\Gamma_1 : \mathbb{F} \to \mathbb{F}^m$ be the degree-h polynomial such that for all $a \in [h]$ it holds that $\Gamma_1(\sigma_{1,a}) = q_{0,0^m,a}$.[48] Now, for $j = 2, ..., i$, we define $\Gamma_j : \mathbb{F} \to \mathbb{F}^m$ as follows.

  - For every $r_{j-1} \in \mathbb{F}$, let $\Gamma_j^{(r_{j-1})} : \mathbb{F} \to \mathbb{F}^m$ be the degree-h polynomial such that for all $a \in [h]$ it holds that $\Gamma_j^{(r_{j-1})}(\sigma_{1,a}) = q_{j-1,\Gamma_{j-1}(r_{j-1}),a}$.

[47] Actually, in the first round the prover knows the points in advance, but it is instructive (for further rounds) to think of the verifier as sending these points to the prover.

[48] We think of 0^m as the index of the output gate of $C_n(x)$ in the topmost layer.

– Let $\Gamma_j \colon \mathbb{F} \to \mathbb{F}^m$ be the polynomial of degree $K \cdot h$ such that for all $(k,a) \in [K] \times [h]$ it holds that $\Gamma_j(\sigma_{k,a}) = \Gamma_j^{(r_{j-1}^{(k)})}(\sigma_{1,a})$.

- **The correct polynomial.** For $i = 1$, the correct polynomial is $p_1 \colon \mathbb{F} \to \mathbb{F}$ such that $p_1(r) = \hat{\alpha}_1(\Gamma_1(r))$. For $i \in \{2,...,R\}$, and any partial transcript of the form $\pi_{\leq i-1} = (x, \pi_1, ..., \hat{\pi}_{i-1})$, and $r_{i-1} \in \mathbb{F}$, the correct polynomial for r_{i-1} is $p_i \colon \mathbb{F} \to \mathbb{F}$ such that $p_i(r) = \hat{\alpha}_i(\Gamma_i^{(r_{i-1})}(r))$. Note that $\deg(p_i) \leq \Delta \cdot h$.

- **Consistency.** Consider a partial transcript $\pi_{\leq 1} = (x, \pi_1)$. We say that π_1 is consistent with $x = (x_0, b)$ if $b = \mathsf{USR}^{\pi_1(\sigma_{1,1}),...,\pi_1(\sigma_{1,h})}(0, 0^m)$, where the oracle notation means that the h queries of USR are answered by $\pi_1(\sigma_{1,1}), ..., \pi_1(\sigma_{1,h})$, in order.

  Now, consider a message $\hat{\pi}_i = (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i')$.

  1. We say that $\hat{\pi}_i$ is self-consistent if for all $k \in [K]$ and $a \in [h]$ it holds that $\pi_i'(\sigma_{k,a}) = \pi_i^{(k)}(\sigma_{1,a})$.
  2. Fix any subsequent *single* challenge $r_i \in \mathbb{F}$ and *single* response π_{i+1}. We say that π_{i+1} is consistent with $\hat{\pi}_i$ at r_i if $\pi_i'(r_i) = \mathsf{USR}^{\pi_{i+1}(\sigma_{1,1}),...,\pi_{i+1}(\sigma_{1,h})}(i, \Gamma_i(r_i))$.[49]
  3. Fix K challenges $\hat{r}_i = \left(r_i^{(1)}, ..., r_i^{(K)}\right)$ and $K+1$ responses $\hat{\pi}_{i+1} = \left(\pi_{i+1}^{(1)}, ..., \pi_{i+1}^{(K)}, \pi_{i+1}'\right)$. We say that $\hat{\pi}_{i+1}$ is consistent with $\hat{\pi}_i$ at $\hat{r}_i$ if for all $k \in [K]$ it holds that $\pi_{i+1}^{(k)}$ is consistent with $\hat{\pi}_i$ at $r_i^{(k)}$.

  For $i \in \{2,...,R\}$ and any partial transcript $\pi_{\leq i} = \langle x, \pi_1, ..., \hat{\pi}_{i-1}, r_{i-1}, \pi_i \rangle$, we say that $\pi_{\leq i}$ is a consistent transcript if π_1 is consistent with x, and for all $j \in \{2,...,i-1\}$ it holds that $\hat{\pi}_j$ is self-consistent and is consistent with $\hat{\pi}_{j-1}$ at $\hat{r}_{j-1}$ (for $j = 2$ we require consistency with π_1 at $\hat{r}_1$), and also π_i is consistent with $\hat{\pi}_{i-1}$ at r_{i-1}. We say that a complete transcript $\pi_{\leq R} = \langle x, \pi_1, \hat{r}_1, ..., \hat{r}_{R-1}, \hat{\pi}_R \rangle$ is a consistent transcript if if π_1 is consistent with x, and for all $j \in \{2,...,R\}$ it holds that $\hat{\pi}_j$ is self-consistent and is consistent with $\hat{\pi}_{j-1}$ at $\hat{r}_{j-1}$.

Defining the promise problem. The promise problem consists of transcripts of valid form, i.e. messages are of the format defined above. We separately define the cases of $i = 1$ and of partial transcripts with $i \in \{2,...,R\}$.

1. **The first round.** Let $\pi_{\leq 1} = \langle x, \pi_1 \rangle$. We say that $\pi_{\leq 1} \in \mathsf{Y}$ if $C_n(x_0) = b$ and π_1 is consistent with x; and we say that $\pi_{\leq 1} \in \mathsf{N}$ otherwise.

2. **A generic round.** For $i \in \{2,...,R\}$, let $\pi_{\leq i} = \langle x, \pi_1, \hat{r}_1, ..., r_{i-1}, \pi_i \rangle$ be a partial transcript. We say that $\pi_{\leq i} \in \mathsf{Y}$ if π_i is the correct polynomial for r_{i-1} and the transcript is consistent. We say that $\pi_{\leq i} \in \mathsf{N}$ if π_i disagrees with the correct polynomial for r_{i-1} on an element in $\{\sigma_{1,a}\}_{a \in [h]}$ or if the transcript is inconsistent.

[49] When $i = 1$, the message π_1 is a single polynomial rather than a tuple. We extend the definition to this case in the obvious way: π_2 is consistent with π_1 if $\pi_1(r_1) = \mathsf{USR}^{\pi_2(\sigma_{1,1}),...,\pi_2(\sigma_{1,h})}(1, \Gamma_1(r_1))$.

7.1.4.3 Analysis: Completeness and soundness

We now argue completeness and soundness of Π, meeting the specifications in Definition 7.2. We do so separately for the first round, for a generic round, and for the last round and final verification procedure.

For convenience, let us recall part of Definition 7.2, referring to a partial transcript $\pi_{\leq i} = \left\langle x, \pi_1, (r_1^{(1)}, ..., r_1^{(K)}), (\pi_2^{(1)}, ..., \pi_2^{(K)}, \pi_2'), ..., (\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi_i') \right\rangle$. We say that:

$$\pi_{\leq i} \text{ is good if } \Pr_{r_i}[\exists \pi_{i+1} : \langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{Y}] = 1$$
$$\pi_{\leq i} \text{ is doomed if } \Pr_{r_i}[\exists \pi_{i+1} : \langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{N}] \geq 1 - 1/3R \,.$$

The first round $i = 1$. For completeness, assume that $x \in \mathsf{Y}_0$. The honest prover sends the polynomial $\pi_1 : \mathbb{F} \to \mathbb{F}$ such that $\pi_1(r) = \hat{\alpha}_1(\Gamma_1(r))$. To see that $\langle x, \pi_1 \rangle$ is good, note that for any $r_1 \in \mathbb{F}$ sent by the verifier, the honest prover can send the correct polynomial π_2 for r_1, in which case $\langle x, \pi_1, r_1, \pi_2 \rangle \in \mathsf{Y}$.[50]

For soundness, we want to show that if $x \in \mathsf{N}_0$, then for every π_1 sent by the prover it holds that $\langle x, \pi_1 \rangle$ is doomed. To see this, note that if π_1 isn't consistent with x then any continuation of the transcript won't be consistent (i.e., regardless of r_1, π_2); thus, we can assume that $C_n(x_0) \neq b = \mathsf{USR}^{\pi_1(\sigma_{1,1}),...,\pi_1(\sigma_{1,h})}(0, 0^m)$. Hence, there is $a \in [h]$ such that π_1 disagrees with the correct polynomial p_1 at $\sigma_{1,a}$. Since π_1 and p_1 are of degree at most $\Delta \cdot h = h^2 \cdot \text{polylog}(T) = \tilde{O}(T^{\gamma/3})$ and $|\mathbb{F}| = p = \Theta(T^{c_0 \cdot \gamma})$, we have that

$$\Pr_{r_1 \in \mathbb{F}}[p_1(r_1) \neq \pi_1(r_1)] > 1 - T^{-(c_0 - 1/3 - o(1)) \cdot \gamma} > 1 - 1/3R \,,$$

where we relied on the fact that $R = (d/\gamma) \cdot \log(T) < d \cdot \log^2(T)$ and on our hypothesis that $d \leq T^{c \cdot \gamma}$, for $c < c_0 - 1/3 - o(1)$.

Condition on any such choice of r_1, and consider any answer π_2 by the prover. If π_2 disagrees with the correct polynomial for r_1 on $\sigma_{1,a}$ for some $a \in [h]$, then by definition $\pi_{\leq 2} = \langle x, \pi_1, r_1, \pi_2 \rangle \in \mathsf{N}$. Otherwise, relying on the facts that $p_1(r_1) = \mathsf{USR}^{\pi_2(\sigma_{1,1}),...,\pi_2(\sigma_{1,h})}(1, \Gamma_1(r_1))$ and $p_1(r_1) \neq \pi_1(r_1)$, we have that π_2 is not consistent with π_1 at r_1, and hence again we have $\pi_{\leq 2} \in \mathsf{N}$.

A generic round $i \in \{2, ..., R-1\}$: Completeness. Let $\pi_{\leq i-1} = \langle x, \pi_1, \hat{r}_1, ..., \hat{\pi}_{i-1} \rangle$ be a partial transcript, and let $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$ and $\pi_i^{(1)}, ..., \pi_i^{(K)}$ be such that for all $k \in [K]$ it holds that $(\pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)}) \in \mathsf{Y}$. Our goal is to show that the honest prover can send π_i' such that for any $r_i \in \mathbb{F}$ there is π_{i+1} satisfying $(\pi_{\leq i-1}, \hat{\pi}_i, r_i, \pi_{i+1}) \in \mathsf{Y}$.

The honest prover sends the polynomial $\pi_i'(r) = \hat{\alpha}_i(\Gamma_i(r))$, and given any challenge $r_i \in \mathbb{F}$, the prover sends the correct polynomial π_{i+1} for r_i. Note that:

1. $\hat{\pi}_i$ is consistent with $\hat{\pi}_{i-1}$ at $\hat{r}_{i-1}$, which follows from the fact that $(\pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)}) \in \mathsf{Y}$ for all $k \in [K]$.

[50] Recall that r_1 and π_2 are a mental experiment, from the definition of good transcripts. In the actual proof system, the verifier will send K challenges in the next round.

2. π_{i+1} is consistent with $\hat{\pi}_i$ at r_i, by the definitions of π_{i+1} and of π'_i.

3. $\hat{\pi}_i$ is self-consistent. This follows from the definitions of π'_i and Γ_i and from the fact that $(\pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)}) \in \mathsf{Y}$ for all $k \in [K]$.

A generic round $i \in \{2, ..., R-1\}$: Soundness. Let $\pi_{\leq i-1} = \langle x, \pi_1, \hat{r}_1, ..., \hat{\pi}_{i-1} \rangle$ and $r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}$ and $\pi_i^{(1)}, ..., \pi_i^{(K)}$, and fix $k \in [K]$ such that $\pi_{\leq i}^{(k)} = \left\langle \pi_{\leq i-1}, r_{i-1}^{(k)}, \pi_i^{(k)} \right\rangle \in \mathsf{N}$. Our goal is to show that for any π'_i sent by the prover, the partial transcript $\pi_{\leq i} = \left\langle \pi_{\leq i-1}, \left(r_{i-1}^{(1)}, ..., r_{i-1}^{(K)} \right), \left(\pi_i^{(1)}, ..., \pi_i^{(K)}, \pi'_i \right) \right\rangle$ will be doomed.

If consistency is violated in $\pi_{\leq i}^{(k)}$, then consistency is also violated in $\pi_{\leq i}$ and in any subsequent transcript. Thus, we only need to handle the case where $\pi_{\leq i}^{(k)}$ is consistent and $\pi_i^{(k)}$ disagrees with the correct polynomial on $\sigma_{1,a}$ for some $a \in [h]$.

Fix any message π'_i from the prover. If $\hat{\pi}_i$ is not self-consistent, we are done. Otherwise, we have $\pi'_i(\sigma_{k,a}) = \pi_i^{(k)}(\sigma_{1,a})$ for all $a \in [h]$.

Denote the correct polynomial for $r_i^{(k)}$ by $p_i^{(k)}$, and let $p'_i(r) = \hat{\alpha}_i(\Gamma_i(r))$. Observe that for every $a \in [h]$ it holds that $p'_i(\sigma_{k,a}) = p_i^{(k)}(\sigma_{1,a})$. Hence, there exists $a \in [h]$ such that $\pi'_i(\sigma_{k,a}) \neq p'_i(\sigma_{k,a})$.[51] And since p'_i and π'_i are distinct polynomials of degree $\Delta \cdot K \cdot h \leq \tilde{O}(T^{\gamma/3}) \cdot K$, we have

$$\Pr_{r_i \in \mathbb{F}} \left[p'_i(r_i) \neq \pi'_i(r_i) \right] \geq 1 - K \cdot T^{-(c_0 - 1/3 - o(1)) \cdot \gamma} \geq 1 - 1/3R \,,$$

where we relied on the fact that $R < d \cdot \log^2(T)$ and on our hypothesis that $K \leq T^{c \cdot \gamma}/d$, for $c < c_0 - 1/3 - o(1)$.

Fixing any choice of r_i such that $p'_i(r_i) \neq \pi'_i(r_i)$, we show that for all π_{i+1} it holds $\langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{N}$. If π_{i+1} disagrees with the correct polynomial on $\{\sigma_{1,a}\}_{a \in [h]}$, we are done. Otherwise, $p'_i(r_i) = \mathsf{USR}^{\pi_{i+1}(\sigma_{1,1}), ..., \pi_{i+1}(\sigma_{1,h})}(i, \Gamma_i(r_i))$. But since $p'_i(r_i) \neq \pi'_i(r_i)$, by the definition of consistency we have that π_{i+1} is inconsistent with $\hat{\pi}_i$ at r_i. Hence, in this case also $\langle \pi_{\leq i}, r_i, \pi_{i+1} \rangle \in \mathsf{N}$.

The final round $i = R$. Let $\pi_{\leq R-1}$ be the transcript going into the final round R, and consider K challenges $r_{R-1}^{(1)}, ..., r_{R-1}^{(K)}$ and K responses $\pi_R^{(1)}, ..., \pi_R^{(K)}$. Replicating the analysis for a generic round,

- If $\left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{Y}$ for all $k \in [K]$, then the prover can send $\pi'_R(r) = \hat{\alpha}_R(\Gamma_R(r))$, in which case the complete transcript is consistent.
- If $\left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{N}$ for some $k \in [K]$, then either consistency is violated, or $\pi_R^{(k)}$ disagrees with the correct polynomial on $\sigma_{1,a}$ for some $a \in [h]$.

Now, let $\pi_{\leq R} = \langle x, \pi_1, \hat{r}_1, ..., \hat{\pi}_R \rangle$ be a complete transcript. The verifier first checks that $\pi_{\leq R}$ is consistent, rejecting if it finds inconsistencies. Then, the verifier accepts if and only if for all $k \in [K]$ and all $a \in [h]$ it holds that $\pi_R^{(k)}(\sigma_{1,a}) = p_R^{(k)}(\sigma_{1,a})$, where $p_R^{(k)}$ is the correct polynomial

[51] This is because for all $a \in [h]$ we have that $\pi'_i(\sigma_{k,a}) = \pi_i^{(k)}(\sigma_{1,a})$ and $p'_i(\sigma_{k,a}) = p_i^{(k)}(\sigma_{1,a})$, but by our assumption $\pi_i^{(k)}$ and $p_i^{(k)}$ disagree on $\sigma_{1,a}$ for some $a \in [h]$.

for $r_{R-1}^{(k)}$. (In the next section we will analyze the verifier's complexity, including the complexity of computing $p_R^{(k)}$.) Thus, when $\left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{Y}$ for all $k \in [K]$ and the prover sends the correct π'_R, the verifier accepts with probability one; and when $\left\langle \pi_{\leq R-1}, r_{R-1}^{(k)}, \pi_R^{(k)} \right\rangle \in \mathsf{N}$ for some $k \in [K]$, the verifier rejects with probability one (regardless of the π'_R that the prover sends).

7.1.4.4 Analysis: Prover and verifier complexity

Lastly, we bound the complexity of the honest prover, of the next-prover-message function (as defined in Definition 7.2), and of the verifier. Our analysis will rely on the following claim, which refers to computing the curves $\Gamma_1, ..., \Gamma_R$.

Claim 7.7.1. *For each $i = 1, \ldots, R$, given $\pi_{\leq i} = \left\langle x, \pi_1, ..., (r_{i-1}^{(1)}, ..., r_{i-1}^{(K)}) \right\rangle$, the polynomial Γ_i can be constructed in time $R \cdot h^{2c'_0} \cdot K^2 \cdot m \cdot \mathrm{polylog}(|\mathbb{F}|)$. The same claim holds for the polynomial $\Gamma_i^{(r_{i-1}^{(k)})}$ for any $k \in [K]$.*

Proof. We iteratively construct Γ_j for $j = 1, ..., i$. For the base case $j = 1$, we simulate $\mathsf{USR}(0, 0^m)$ in order to compute its queries $\{q_{0,0^m,a}\}_{a\in[h]}$. (Recall that USR is non-adaptive.) Then, for each of the output coordinates $u \in [m]$ of Γ_1, we use fast interpolation to find the coefficients of the degree-h polynomial $r \mapsto \Gamma_1(r)_u$.[52]

Now, let $j > 1$, and assume that we have a representation of Γ_{j-1}. We first run $\mathsf{USR}(j-1, \Gamma_{j-1}(r_{j-1}^{(k)}))$ for every $k \in [K]$, to obtain the query locations $\left\{q_{j-1,\Gamma_{j-1}(r_{j-1}),a}\right\}_{k\in[K],a\in[h]}$, and then for each $u \in [m]$ use fast interpolation to find the coefficients of the degree-$(K \cdot h)$ polynomial $r \mapsto \Gamma_j(r)_u$ (where Γ_j is such that $\Gamma_j(\sigma_{k,a}) = q_{j-1,\Gamma_{j-1}(r_{j-1}^{(k)}),a}$ for all $k \in [K], a \in [h]$).

The overall time complexity is

$$R \cdot \left(\underbrace{K \cdot h^{c'_0}}_{\mathsf{USR}} + \underbrace{(K \cdot h)^2}_{\text{evaluating } \Gamma_j \text{ at } K \cdot h \text{ inputs}} + m \cdot \underbrace{\tilde{O}(K \cdot h)}_{\text{FFT}} \right) \cdot \mathrm{polylog}(|\mathbb{F}|) \,,$$

which is bounded by $R \cdot h^{2c'_0} \cdot K^2 \cdot m \cdot \mathrm{polylog}(|\mathbb{F}|)$. The proof of the claim about constructing $\Gamma_i^{(r_{i-1}^{(k)})}$ is essentially identical. □

Time complexity of the honest prover. To compute π_1, the honest prover constructs Γ_1 using Claim 7.7.1, evaluates $\hat{\alpha}_1$ at points $S_1 = \{\Gamma_1(\sigma_{1,a,b})\}_{a\in[h],b\in[\Delta]} \cup \{\Gamma_1(\sigma_{1,1,\Delta+1})\}$, and lets $\pi_1 \colon \mathbb{F} \to \mathbb{F}$ be the unique degree-$(\Delta \cdot h)$ polynomial that evaluates to $\hat{\alpha}_1(s)$ for every $s \in S$. Since $\hat{\alpha}_1 \circ \Gamma_1$ is of degree $\Delta \cdot h$, we have that $\pi_1 \equiv \hat{\alpha}_1(\Gamma_1(r))$.

Now, in each round $i \in [R-1]$, given $\bar{\pi}_i = (x, \pi_1, ..., \hat{\pi}_i)$ and $r_i^{(1)}, ..., r_i^{(K)}$, the honest prover constructs $\Gamma_i^{(r_{i-1}^{(1)})}, ..., \Gamma_i^{(r_{i-1}^{(K)})}, \Gamma_i$ using Claim 7.7.1. Then, for every $k \in [K]$ it evaluates $\hat{\alpha}_i$ at points $S_i^{(k)} = \left\{\Gamma_i^{(r_{i-1}^{(k)})}(\sigma_{1,a,b})\right\}_{a\in[h],b\in[\Delta]} \cup \left\{\Gamma_i^{(r_{i-1}^{(k)})}(\sigma_{1,1,\Delta+1})\right\}$, and uses fast interpolation to construct the

[52] Recall that Γ_1 is the unique polynomial of degree h that for every $a \in [h]$ satisfies $\Gamma_1(\sigma_{1,a}) = q_{0,0^m,a}$.

unique degree-$(\Delta \cdot h)$ polynomial $\pi_i^{(k)} : \mathbb{F} \to \mathbb{F}$ that agrees with $\hat{\alpha}_i$ on $S_i^{(k)}$. Similarly, it constructs the unique degree-$(\Delta \cdot h \cdot K)$ polynomial $\pi_i' : \mathbb{F} \to \mathbb{F}$ that agrees with $\hat{\alpha}_i$ on $\{\Gamma_i(\sigma_{k,a,b})\}_{k\in[K],a\in[h],b\in[\Delta]} \cup \{\Gamma_i(\sigma_{1,1,\Delta+1})\}$.

The time complexity of a generic step is higher than that of computing π_1, and it is bounded by (omitting a multiplicative polylog$(|\mathbb{F}|)$ factor for readability)

$$K \cdot \underbrace{R \cdot h^{2c_0'} \cdot K^2 \cdot m}_{\text{Claim 7.7.1}} + \underbrace{\tilde{O}(K \cdot \Delta \cdot h)}_{\text{FFTs}} + \underbrace{T^{c_0'} \cdot K \cdot \Delta \cdot h}_{\text{computing } \hat{\alpha}_i} ,$$

which is at most $T^{c_0'} \cdot K^3 \cdot \Delta \cdot h^{2c_0'} \cdot d' \cdot m^2 \cdot \text{polylog}(|\mathbb{F}|)$. In the first round, the prover's message is of length $O(\Delta \cdot h \cdot \log(|\mathbb{F}|))$, and in subsequent rounds it sends $K+1$ messages of total length at most $O(K \cdot h \cdot \Delta) \cdot \log(|\mathbb{F}|)$.

Next-prover-message space complexity. Given a partial transcript $\pi_{\le i} = \langle x, \pi_1, ..., \hat{\pi}_i, r_i \rangle$, we want to decide whether or not there exists π_{i+1} such that $\langle \pi_{\le i}, \pi_{i+1} \rangle \notin \mathsf{N}$. Recall that $\pi_{\le i+1} = \langle \pi_{\le i}, \pi_{i+1} \rangle \notin \mathsf{N}$ if $\pi_{\le i+1}$ is consistent and agrees with the correct polynomial p_{i+1} for r_i on all elements in $\{\sigma_{1,a}\}_{a\in[h]}$.

To do so, we first check for consistency of $\pi_{\le i}$ (if it is not consistent, we reject). When $\pi_{\le i}$ is consistent, the question reduces to checking whether there exists π_{i+1} that is consistent with $\hat{\pi}_i$ at r_i and that agrees with p_{i+1} on $(\sigma_{1,a})_{a\in[h]}$.

Claim 7.7.2. *There exists π_{i+1} such that $\pi_{\le i+1} \notin \mathsf{N}$ if and only if*

$$\pi_i'(r_i) = \mathsf{USR}^{p_{i+1}(\sigma_{1,1}),\ldots,p_{i+1}(\sigma_{1,h})}(i, \Gamma_i(r_i)) . \tag{7.2}$$

Proof. If such π_{i+1} exists, then by the consistency requirement it satisfies

$$\pi_i'(r_i) = \mathsf{USR}^{\pi_{i+1}(\sigma_{1,1}),\ldots,\pi_{i+1}(\sigma_{1,h})}(i, \Gamma_i(r_i)) .$$

Also, since π_{i+1} agrees with p_{i+1} on $(\sigma_{1,a})_{a\in[h]}$, Eq. (7.2) should hold. Thus, if Eq. (7.2) does not hold, there is no suitable π_{i+1}.

On the other hand, if Eq. (7.2) holds, then any π_{i+1} that agrees with p_{i+1} on $\{\sigma_{1,a}\}_{a\in[h]}$ is consistent with π_i' on r_i. □

Recall that USR is computable in space $O(\log T)$, and we assume that the (x, g) to the value of the g^{th} gate in $C_n(x)$ can be done in space S. By the "furthermore" part of Proposition 7.7, we can compute $(i, \vec{w}) \mapsto \hat{\alpha}_i(\vec{w})$ in space $O(S + \log T)$. We will also rely on the following space-efficient way of computing each coordinate of $\Gamma_i(r_i)$:

Claim 7.7.3. *Each output coordinate of $\Gamma_i(r_i)$ can be computed in space $O(d' \cdot m \cdot \log(T \cdot K))$.*

Proof. For each $u \in [m]$, we compute the u^{th} coordinate of $\Gamma_i(r_i)$ as

$$\sum_{(k,h)\in[K]\times H} \delta_{k,a}(r_i) \cdot (q_{i-1,\Gamma_{i-1}(r_{i-1}),a})_u \tag{7.3}$$

where $\delta_{k,h}(r) = \prod_{(k',h')\in([K]\times H)\setminus\{k,h\}} \frac{\sigma_{k',h'} - r}{\sigma_{k',h'} - \sigma_{k,h}}$. (Note that the polynomial in Eq. (7.3) is of degree $K \cdot h - 1$ and it agrees with the u^{th} coordinate of Γ_i on the $K \cdot h$ points $\sigma_{k,h}$.)

To compute Eq. (7.3) we compute $\Gamma_{i-1}(r_{i-1})$, then iterate over the pairs (k,h), and for each pair we simulate USR on input $(i-1,\Gamma_{i-1}(r_{i-1}))$ to obtain the u^{th} coordinate of the a^{th} query. The size of $\Gamma_{i-1}(r_{i-1})$ is $m\cdot\log(|\mathbb{F}|)\leq O(\log T)$, and USR runs in space $O(\log T)$; hence, computing $\Gamma_i(r_i)_u$ reduces in space $O(\log T+\log(K\cdot h))\leq O(\log(T\cdot K))$ to computing $\Gamma_{i-1}(r_{i-1})$.

Using space-efficient composition with R levels of recursion, the space complexity of computing each output coordinate is $O(R\cdot\log(T\cdot K))$ □

To check whether Eq. (7.2) holds we compute $\Gamma_i(r_i)$, simulate USR on input $(i,\Gamma_i(r_i))$, and compare the result to $\pi'_i(r_i)$ (recall that π'_i and r_i are part of our input, so computing $\pi'_i(r_i)$ is straightforward). The only missing thing is answering the queries of USR. We store a counter of $a\in[h]$, and whenever USR makes an oracle query we answer with $p_{i+1}(\sigma_{1,a})$ and increment a, where p_{i+1} is the correct polynomial for r_i.

Recall that $p_{i+1}(\sigma_{1,a})=\hat{\alpha}_{i+1}(\Gamma_{i+1}^{(r_i)}(\sigma_{1,a}))$, and that we can compute $\hat{\alpha}_{i+1}$ in space $O(S+\log T)$. Thus, it is just left to bound the space-complexity of computing $\Gamma_{i+1}^{(r_i)}(\sigma_{1,a})=q_{i,\Gamma_i(r_i),a}$. This can be done by simulating USR and giving it virtual access to the input $(i,\Gamma_i(r_i))$, relying on Claim 7.7.3.

The overall space complexity of the procedure is dominated by S and by the space complexity of computing $\Gamma_i(r_i)$, so it is overall bounded by

$$O(d'\cdot m\cdot\log(T\cdot K)+S)\ .$$

(Recall that the input is of length $|\pi_{\leq i}|\leq O(R\cdot K\cdot\Delta\cdot h\cdot\log(|\mathbb{F}|))$, and thus $\log(|\pi_{\leq i}|)$ is smaller than the expression above.)

Complexity of the verifier. Recall that, given a complete transcript, the verifier checks its consistency, then accepts if and only if for all $k\in[K]$ and all $a\in[h]$ it holds that $\pi_R^{(k)}(\sigma_{1,a})=p_R^{(k)}(\sigma_{1,a})$, where $p_R^{(k)}$ is the correct polynomial for $r_{R-1}^{(k)}$.

Self-consistency of each of the prover messages can be checked in time $K\cdot h\cdot\Delta$, so checking all of them can be done in time $K\cdot h^2\cdot d'\cdot m\cdot\text{polylog}(T)$. Consistency between each subsequent pair of messages can be checked in time $K\cdot h^{c'_0}+h\cdot\Delta$, and thus testing all subsequent pairs can be done in time $K\cdot h^{2c'_0}d'\cdot m\cdot\text{polylog}(T)$.

Finally, the verifier enumerates over $k\in[K]$ and $a\in[h]$. Recall that $p_R^{(k)}(\sigma_{1,a})=\hat{\alpha}_R(\Gamma_R^{(r_{R-1}^{(k)})}(\sigma_{1,a}))$. The verifier constructs the curve $\Gamma_R^{(r_{R-1}^{(k)})}$ using Claim 7.7.1, and computes $\vec{w}_{a,k}=\Gamma_R^{(r_{R-1}^{(k)})}(\sigma_{1,a})\in\mathbb{F}^m$. To evaluate $\hat{\alpha}_R$ at $\vec{w}_{a,k}$ the verifier simulates USR, answering its queries to the bottom layer using its input x and the base layer algorithm. The time complexity of this step is

$$K\cdot h\cdot\left(R\cdot h^{2c'_0}\cdot K^2\cdot m\cdot\text{polylog}(|\mathbb{F}|)+h^{2c'_0}\cdot\max\{n,h\}\right)\ ,$$

which is at most $K^3\cdot h^{3c'_0}\cdot\max\{n,h\}\cdot d'\cdot m^2\cdot\text{polylog}(T)$.

7.2 List-CIHFs suffice for NIZK

In this section we prove that, assuming sub-exponentially secure one-way functions, targeted HSGs that fool appropriate distinguishers (equivalently, list-CIHFs that avoid appropriate relations) suffice to construct zaps, NIWI and NIZK protocols. That is:

Theorem 7.8 (list-CIHFs suffice for zaps and NIWI)**.** *Assume that:*

1. *There are sub-exponentially secure one-way functions.*
2. *For a sufficiently small $\epsilon > 0$, there is a $(1/2, n^{-\omega(1)})$-targeted HSG for $\mathcal{DTIME}[n]^{\mathsf{SAT}}$ over all distributions samplable in polynomial time such that on inputs of length n, the targeted HSG runs in time $\mathrm{poly}(n)$ and prints $\mathrm{poly}(k)$ strings of length $k = k(n) = n^{\epsilon}$.*[53]

Then, every $\mathcal{NP}$ relation R has a zap argument system. Assuming in addition a sub-exponentially secure non-interactive commitment scheme, every $\mathcal{NP}$ relation R has a NIWI *argument system.*

Recall that, combining the concluded zap in Theorem 7.8 with Lemma 3.7, the former also yields NIZK arguments.

Similarly to Theorem 7.4, the choice of parameters for the targeted HSG in Theorem 7.8 is motivated by known constructions. Specifically, a targeted HSG with such parameters was constructed in [CT21; CT23] from hardness over all efficiently samplable distributions.

The rest of this section is devoted to the proof of Theorem 7.8. Let $R \in \mathcal{NP}$. We assume wlog that membership in R can be verified in quasi-linear time. (To see that this is wlog, recall that the $\mathcal{NP}$-complete problem circuit satisfiability has this feature, and a zap/NIWI for it implies a zap/NIWI for all of $\mathcal{NP}$.) Throughout this proof we use n to denote the length of instances for R and, jumping ahead, we will denote by N the input length to the targeted HSG.

Technical ingredients and parameter setting. We first instantiate the ingredients that are used in the construction:

1. **(Commitment scheme.)** Let commit be a $2^{\lambda^{\zeta}}$-secure non-interactive commitment scheme in the CRS model, for $\zeta > 0$. Let $c_0 > 1$ be such that commit runs in time at most n^{c_0}, when instantiated with security parameter $\lambda = n$ (note in particular that this means that the commitments have length at most n^{c_0}).
2. **(Targeted HSG.)** Let $c_1 > 1$ be such that the targeted HSG, when instantiated with input length N and output length k, prints k^{c_1} strings.
3. **(ZK PCP.)** Let $c_2 > 1$ be a constant such that when instantiating Theorem 3.10 with any parameter $q_{\max}$, the alphabet size is at most $2^{(n \cdot q_{\max})^{c_2}}$ and the running-time of the verifier is at most $(n \cdot q_{\max})^{c_2}$. We instantiate Theorem 3.10 with parameter $q_{\max} = n^{\mu}$, where $\mu < \zeta/2$. By our choice, we have

$$2^{n^{\zeta}} > O(q_{\max})^{q_{\max}} \,. \tag{7.4}$$

   Note that the PCP has soundness error $\delta_S \leq 1/2$, proof length $\ell = O(q_{\max})$, query complexity $q = O(\sqrt{q_{\max}})$, randomness complexity $r = O(\log \ell)$, verifier time $T_V = (n \cdot q_{\max})^{c_2}$ and prover time $\mathrm{poly}(n, q_{\max})$.

[53] The meaning of "$\mathcal{DTIME}[n]^{\mathsf{SAT}}$" here is that for every algorithm D that gets input $z \in \{0,1\}^n$ and random coins $r \in \{0,1\}^k$ and runs in linear time $O(n)$ while making oracle queries to SAT, and every polynomial-time samplable distribution $\mathbf{z} = \{\mathbf{z}_n\}_{n \in \mathbb{N}}$, the algorithm $D(z, \cdot)$ does not distinguish between a uniformly random choice of r and a pseudorandom choice of r from the output-list of the generator on z (except with negligible probability over $z \sim \mathbf{z}_n$).

4. **(Input and output lengths for the targeted HSG.)** Jumping ahead, we will be instantiating the targeted HSG on inputs of length $N = n^{c_3}$, where $c_3 \geq 1$ is chosen such that $N \geq \Omega(q_{\max} \cdot \log(|\Sigma|) \cdot n^{c_0})$ and $c_3 \geq \max(c_0 + 2, 2\mu \cdot c_0 \cdot c_2, 2(1+\mu) \cdot c_2)$. For $\epsilon < \mu/(2c_1 \cdot c_3)$, the output length of the targeted HSG will be $k = k(N) = N^{\epsilon}$ bits. Note that

$$q_{\max} \geq k^{c_1} \cdot q \tag{7.5}$$

and[54]

$$k \geq r \,. \tag{7.6}$$

The construction. We denote by $V_{zkPCP}(x, a; \rho)$, the output of the PCP verifier V_{zkPCP} given $x \in \{0,1\}^n$ as its main input, using random coins $\rho \in \{0,1\}^r$ and when given $a \in \Sigma^q$ as the answers to its queries. We denote by $pcp \leftarrow P_{zkPCP}(x, w)$ the proof string generated by the PCP prover given the instance x and corresponding witness w; recall that the prover is randomized, and thus pcp is a random variable.

Construction 7.9. *Consider the non-interactive protocol (P, V), given a fixed input x, witness w (for the prover), and $crs \in \{0,1\}^n$:*

The prover $P(crs, x, w)$:

1. *Generate a zkPCP proof-string $pcp \leftarrow P_{zkPCP}(x, w)$.*
2. *For $i \in [\ell]$, draw a random $r_i \in \{0,1\}^{\mathrm{poly}(n)}$ and generate $c_i = \mathsf{commit}(crs, pcp_i; r_i)$ (i.e., the* commit *algorithm uses r_i as a random string), where $pcp_i \in \Sigma$ denotes the i-th symbol of pcp.*[55]
3. *Generate the list S of strings that the targeted HSG prints given on input $(crs, x, c_1, \ldots, c_\ell)$. (Indeed, the targeted HSG expects an input of length N; by our setting of parameters $N \geq |crs| + |x| + \sum_i |c_i|$, and we pad the input with zeros in case it is shorter than N.) Note that S is of size $|S| = k^{c_1}$; also, each string in S has length k, but we will be using only their r-bit prefixes (note that $r \leq k$ by Eq. (7.6)) and so we refer to the elements of S as having length r.*
4. *For every $\rho \in S$, let $Q_\rho \subseteq [\ell]$ denote the PCP queries that V_{zkPCP} makes given input x and random string ρ.*
5. *The proof-string is defined as $\left((c_i)_{i\in[\ell]}, Q, (pcp_j, r_j)_{j\in Q}\right)$, where $Q = \cup_{\rho\in S} Q_\rho$ and we additionally pad Q so that $|Q| = q_{\max}$ (note that by Eq. (7.5), it holds that $q_{\max} \geq k^{c_1} \cdot q = |S| \cdot q \geq |\cup_{\rho\in S} Q_\rho|$).*

The verifier $V\left(crs, x, \left((c_i)_{i\in[\ell]}, Q, (pcp_j, r_j)_{j\in Q}\right)\right)$:

1. *Generate the list S of strings that the targetd HSG prints on input $(crs, x, c_1, \ldots, c_\ell)$, and confirm that $Q \supseteq \cup_{\rho\in S} Q_\rho$ and $|Q| = q_{max}$.*
2. *For every $j \in Q$, check that $c_j = \mathsf{commit}(crs, pcp_j; r_j)$.*
3. *For every $\rho \in S$, check that $V_{zkPCP}(x, a_\rho; \rho)$ accepts when given as answers $a_\rho = (pcp_j)_{j\in Q_\rho}$.*
4. *Accept if and only if all of the above tests pass.*

[54] Indeed, this is the reason that we needed to derandomize the [HVW22] construction (see Appendix A).

[55] We remark that commit was defined as a bit-commitment scheme and so the actual implementation does a bit-by-bit to the $\log(|\Sigma|)$ bits of each pcp_i.

Analysis. The completeness of Construction 7.9 follows from the perfect completeness of the zkPCP. In more detail, for every (x, w) and crs, the prover can generate pcp such that $V_{zkPCP}(x)$ with oracle access to pcp always accepts, commit correctly to the symbols of pcp, and decommit to the requested queries.

We now argue that Construction 7.9 is sound against efficient adversaries. In fact, we prove adaptive soundness: the construction is sound even when the adversary is allowed to see the crs. The proof relies on the pseudorandomness of the targeted HSG as well as on the binding property of the commitment.

Proposition 7.10 (adaptive soundness)**.** *For every ppt algorithm P^* and every sufficiently large $n \in \mathbb{N}$ it holds that*

$$\Pr_{\substack{crs \leftarrow \{0,1\}^n \\ (x,\alpha) \leftarrow P^*(crs)}} \Big[\big(x \notin L(R)\big) \text{ and } \big(V(crs, x, \alpha) = 1\big) \Big] \leq \mathsf{negl}(n) \,.$$

Proof. Assume there exists a ppt cheating prover P^* such that for infinitely many $n \in \mathbb{N}$, with noticeable probability over $crs \leftarrow \{0,1\}^n$ and $(x, \alpha) \leftarrow P^*(crs)$ it holds that

$$\big(x \notin L\big) \text{ and } \big(V(crs, x, \alpha) = 1\big) \,. \tag{7.7}$$

Fix an input length $n \in \mathbb{N}$, an $x \in \{0,1\}^n \backslash L$, a $crs \in \{0,1\}^n$ for which the binding condition (of Definition 3.8) is satisfied, and (x, α) in the support of $P^*(crs)$ such that $V(crs, x, \alpha) = 1$. We say that (x, α, crs) as above are *good*.

Interpret α as $\alpha = \big((c_i)_{i \in [\ell]}, Q, (pcp_j, r_j)_{j \in Q}\big)$. Let S be the list of strings that the targeted HSG prints on input $(crs, x, c_1, \ldots, c_\ell)$. Since $V(crs, x, \alpha) = 1$, by construction $Q \supseteq \cup_{\rho \in S} Q_\rho$. (Recall that Q_ρ denotes the PCP query locations generated by V_{zkPCP} given input x and randomness ρ.)

By the binding condition of the commitment scheme, for every $i \in [\ell]$ there exists at most one value $pcp_i \in \Sigma$ such that c_i is in the support of $\mathsf{commit}(crs, pcp_i)$. For those cases in which c_i is not in the support of any of the distributions, we set a default value $pcp_i = 0$. Overall, this defines a candidate PCP proof-string $pcp = (pcp_1, \ldots, pcp_\ell)$. Also, since $V(crs, x, \alpha) = 1$, by construction it also holds that $c_j = \mathsf{commit}(crs, pcp_j; r_j)$, for every $j \in Q$.

Since V explicitly checks that V_{zkPCP} accepts for all $\rho \in S$, the fact that V accepts means that for all $\rho \in S$ it holds that $V_{zkPCP}(x, pcp|_{Q_\rho}; \rho) = 1$. On the other hand, by the soundness of the PCP, since $x \notin L$ and $pcp \in \Sigma^\ell$ is a fixed string, it holds that $V_{zkPCP}(x, pcp|_{Q_\rho}; \rho) = 1$ with probability at most $\delta_S \leq 1/2$ (over $\rho \in \{0,1\}^r$).

Consider a distinguisher $D(z, \rho)$, where $z = \big(crs, x, (c_1, \ldots, c_\ell)\big)$, that distinguishes whether ρ is sampled uniformly from S or uniformly at random in $\{0,1\}^r$. Assume momentarily that $D(z, \rho)$ has access to the pcp string as described above. The distinguisher $D(z, \rho)$ outputs the negation of $V_{zkPCP}(crs, x, pcp|_{Q_\rho})$. By the above discussion, $D(z, \rho)$ is a $(1/2)$-dense avoider for the output list of the targeted generator on z. Assuming access to the pcp string, we get that D runs in time $O(T_V) = O(n \cdot q_{max})^{c_2}$.

To get rid of the assumption that D has access to the pcp string hidden inside $(c_1, \ldots, c_\ell)$, we have D first open the commitments using an $\mathcal{NP}$ oracle, as follows. Let $\mathcal{O} : \{0,1\}^n \times \{0,1\}^{n^{c_0}} \to \{0,1\}$ be an oracle for the language $\{(crs, c) : c \in \mathsf{commit}(crs, 1, \cdot)\}$. Note that on inputs (crs, c) of length $n^{1+c_0} < k^{1/3\epsilon}$ the oracle $\mathcal{O}$ is computable in non-deterministic linear time $O(n^{1+c_0})$; in particular, queries to $\mathcal{O}$ can be resolved by making queries of length less than $k^{1/2\epsilon}$ to SAT. Using

$\mathcal{O}$ we can find the symbols underlying $c_1, \ldots, c_\ell$ using $\ell \cdot \log(|\Sigma|)$ queries to $\mathcal{O}$ (i.e., one per bit of the PCP string). Overall, the running time of D is

$$\ell \cdot \log(|\Sigma|) \cdot \tilde{O}\left(n^{1+c_0}\right) + O(n \cdot q_{max})^{c_2} \leq n^{c_0+2+2c_2(1+\mu))} \leq N \ ,$$

while making $\ell \cdot \log(\Sigma)$ oracle queries to SAT.

By our assumption and the binding of the commitment scheme, $z = \left(crs, x, (c_1, \ldots, c_\ell)\right)$, sampled as $crs \leftarrow \{0,1\}^n$ and $\left((c_i)_{i\in[\ell]}, Q, (pcp_j, r_j)_{j\in Q}\right) \leftarrow P^*(crs)$ is *good* with noticeable probability. Hence, with noticeable probability over z sampled as above, the algorithm D is a $(1/2)$-dense avoider for the outputs of the targeted HSG, a contradiction. ■

Finally, we claim that Construction 7.9 satisfies the zero-knowledge condition, i.e. the interaction can be simulated (by an unbounded simulator) without access to the witness.

Proposition 7.11 (Unbounded Simulator Zap)**.** *Construction 7.9 satisfies the unbounded simulator zap condition of Definition 3.4.*

Note that the WI zap property follows from Proposition 7.11 via Lemma 3.6. Since Proposition 7.11 is the last missing piece towards establishing of Theorem 7.8, the rest of the section is devoted to the proof of Proposition 7.11.

The proof is very similar in structure to the proof of zero-knowledge for the construction in Theorem 5.1. Specifically, the unbounded simulator S acts as follows:

Given input $x \in L \cap \{0,1\}^n$ and $crs \in \{0,1\}^n$, the simulator repeats the following procedure at most $t = \ell^{c \cdot q_{\max}}$ times, where $c > 0$ is a constant that will be set below:

1. Select a random set $Q_{sim} \subseteq [\ell]$ of size $|Q_{sim}| = q_{\max}$.
2. Run the zkPCP simulator to generate $pcp \in \Sigma^\ell$, such that $pcp|_{Q_{sim}}$ is distributed identically to an honestly-generated PCP, and for indices $i \notin Q_{sim}$ we set $pcp_i = 0$.
3. For every $i \in [\ell]$, choose a random $r_i \in \{0,1\}^{\mathrm{poly}(n)}$ and generate the commitment $c_i = \mathsf{commit}(crs, pcp_i; r_i)$.
4. Compute the list S of outputs of the targeted generator on input $(crs, x, c_1, \ldots, c_\ell)$, and let $Q = \cup_{\rho\in S} Q_\rho$ (recall that Q_ρ are the queries that the zkPCP verifier makes given input x and random string ρ). Similarly to the verification procedure we use (the same) padding to ensure that $|Q| = q_{\max}$)
5. If $Q \neq Q_{sim}$ then proceed to the next iteration. Otherwise (i.e., $Q_{sim} = Q$), output $\alpha_{sim} = \left((c_i)_{i\in[\ell]}, Q, (pcp_j, r_j)_{j\in Q}\right)$ and abort.

In case all t iterations fail, the simulator outputs $\perp$.

Notation and basic facts. Similarly to the proof of Theorem 1.8, we fix an input x, a witness w, an auxiliary input z, and a crs, and define a sequence of random variables (with respect to the fixed (crs, x, w, z)) that will be useful for the analysis.

- $\xi_{real}(crs, x, w, z)$: Select a random set $Q_{sim} \subseteq [\ell]$ of size $|Q_{sim}| = q_{\max}$. Generate the zkPCP pcp at random using (x, w). For every $i \in [\ell]$, generate the commitment $c_i = \mathsf{commit}(crs, pcp_i; r_i)$, using uniformly random coins r_i. Let S be the output of the targeted HSG on input $(crs, x, c_1, \ldots, c_\ell)$ and let $Q = \cup_{\rho \in S} Q_\rho$ (with padding as above). If $Q \neq Q_{sim}$ output $\perp$, otherwise output $\Big(crs, x, z, \big((c_i)_{i\in[\ell]}, Q, (pcp_j, r_j)_{j\in Q}\big)\Big)$.

- $\xi_{mid}(crs, x, w, z)$: Same as $\xi_{real}(crs, x, w, z)$, except that rather than committing to the symbols of pcp, the commitments are to the symbols of pcp_{mid} which is identical to pcp on coordinates in Q_{sim} and 0 everywhere else.

- $\xi_{sim}(crs, x, w, z)$: Same as $\xi_{real}(crs, x, w, z)$, except that rather than committing to the symbols of pcp, the commitments are to the symbols of pcp_{sim} which is generated by the simulator of the zkPCP with input x and coordinate set Q_{sim} (note that pcp_{sim} is 0 in all coordinates outside of Q_{sim}).

- $\rho_{real}(crs, x, w, z)$: take t independent copies of $\xi_{real}(crs, x, w, z)$ and output the first output that is not $\perp$. In case all copies are $\perp$ then output $\perp$.

- $\rho_{mid}(crs, x, w, z)$: is defined similarly to $\rho_{real}(crs, x, w, z)$ relative to ξ_{mid}.

- $\rho_{sim}(crs, x, w, z)$: is defined similarly to $\rho_{real}(crs, x, w, z)$ relative to ξ_{sim}.

We argue that ρ_{real} is (up to a small error) the real distribution that the verifier sees with the fixed (x, w, crs), that ρ_{sim} is the simulated distribution, and that ξ_{mid} and ξ_{sim} are identically distributed. The proofs are essentially identical to the ones of Claims 5.3 to 5.5, respectively.

Claim 7.12. *For any fixed (crs, x, w, z) such that $(x, w) \in R$, the following pairs of RVs are distributed identically:*

1. *$\rho_{sim}(crs, x, w, z)$ and $\big(crs, x, z, S(crs, x)\big)$.*
2. *$\xi_{sim}(crs, x, w, z)$ and $\xi_{mid}(crs, x, w, z)$.*
3. *$\rho_{real}(crs, x, w, z)$ and $P(crs, x, w)$,[56] up to error 2^{-n} (i.e., the statistical distance between the RVs is at most 2^{-n}).*

Proof. The proof that $\rho_{real}(crs, x, w, z)$ and $P(crs, x, w)$ are close is essentially identical to that of Claim 5.3, relying on a sufficiently large choice of constant $c > 1$ (in the definition of t) so that the probability that $Q_{sim} \neq Q$ in all t iterations of the simulator is at most $(\ell^{-q_{max}})^t \geq 2^{-n}$.

The equivalence of $\xi_{sim}(crs, x, w, z)$ and $\xi_{mid}(crs, x, w, z)$ follows by the zero-knowledge property of the PCP, which asserts that for any fixed choice of $Q = Q_{sim} \subseteq [\ell]$ of size $|Q| \leq q_{max}$, the distribution of the simulated PCP on coordinates Q is identical to the distribution of the actual PCP (corresponding to w) on coordinates Q. We can rely on this property of the zkPCP since $(x, w) \in R$ and since, by Eq. (7.5), it holds that $|Q| \leq |S| \cdot q \leq k^{c_1} \cdot q \leq q_{\max}$.

The equivalence of $\rho_{sim}(crs, x, w, z)$ and $(crs, x, z, S(crs, x))$ follows directly from the definitions of ρ_{sim} and of the simulator S. □

[56] Recall that $P(crs, x, w)$ is the view of the verifier in the real interaction with input x, CRS crs, and when the prover gets witness w.

The security reduction. Let (A_1, A_2) be a pair of ppt adversaries for the unbounded simulator property. As in the proof of Theorem 5.1, we assume without loss of generality that for any (x, w) generated by A_1 it holds that $(x, w) \in R$, and that w is included in the auxiliary information z.

We now denote by (crs, x, w, z) the *distribution* generated by $A_1(1^n)$. Assume toward a contradiction that there exists a polynomial p such that

$$\left|\Pr\left[A_2(crs, x, P(crs, x, w), z) = 1\right] - \Pr\left[A_2(crs, x, Sim(crs, x), z) = 1\right]\right| \geq 1/p(n)\,, \tag{7.8}$$

for infinitely many n. Then:

Claim 7.13. *There exists a* $\mathrm{poly}(t)$*-time algorithm* D' *such that:*

$$\left|\Pr\left[D'\left(crs, x, z, \xi_{real}(crs, x, w, z)\right) = 1\right] - \Pr\left[D'\left(crs, x, z, \xi_{mid}(crs, x, w, z)\right) = 1\right]\right| \geq \frac{1}{2 \cdot t \cdot p(n)},$$

for infinitely many n*.*

Proof. We first argue that there is a $\mathrm{poly}(t)$-time algorithm D' such that

$$\left|\Pr\left[D'\left(crs, x, z, \xi_{real}(crs, x, w, z)\right) = 1\right] - \Pr\left[D'\left(crs, x, z, \xi_{sim}(crs, x, w, z)\right) = 1\right]\right| \geq \frac{1}{2 \cdot t \cdot p(n)},$$

for infinitely many n. The proof of this fact is essentially identical to the proof of Claim 5.6: We use a uniform hybrid argument, and rely on the fact that D' can generate the RV ξ_{real} using the witness w. The claim follows since ξ_{mid} and ξ_{sim} are distributed identically (by Claim 7.12.) □

Consider an adversary D'' that gets either $2\ell \cdot \log(|\Sigma|)$ bit commitments to 0 or $\ell \cdot \log(|\Sigma|)$ bit commitments to 0 followed by $\ell \cdot \log(|\Sigma|)$ commitments to 1.

The algorithm D'' generates (crs, x, w, z) using A_1. It then generates the proof-string α as in the real interaction (using (crs, x, w)), where for coordinates outside of Q_{sim} it attempts to reconstruct the symbols of pcp as follows. For coordinates within Q_{sim} it generates the commitments to the corresponding symbols of pcp on its own, and for the rest, it uses the bit commitments that it was given as input (working on the assumption that the first batch consists of commitments to 0 and the second to commitments to 1). The algorithm D'' then outputs $D'(crs, x, w, z, \alpha)$, where D' is the distinguisher of Claim 7.13.

Observe that in the first case (all commitments are 0), the distribution that D'' feeds into D' is $\left(crs, x, w, z, \xi_{mid}(crs, x, w, z)\right)$ whereas in the second case (second half are commitments to 1) the distribution is $\left(crs, x, w, z, \xi_{real}(crs, x, w, z)\right)$. Thus, by Claim 7.13 the algorithm D'' distinguishes these two cases with advantage $1/(2 \cdot t \cdot p(n))$ advantage on infinitely many input lengths $n \in \mathbb{N}$.

By another uniform hybrid argument (i.e., over the $2\ell \cdot \log(|\Sigma|)$ commitments, identically to the proof of Theorem 5.1), there is a $\mathrm{poly}(t)$-time adversary that distinguishes a (single) commitment to zero from a commitment to one with advantage $\frac{1}{4 \cdot t \cdot p(n) \cdot \ell \cdot \log(|\Sigma|)} \geq 1/\mathrm{poly}(t)$ on infinitely many input lengths. This contradicts the hiding property of the commitment scheme.

7.3 List-CIHFs from non-batch-computability

In Sections 7.1 and 7.2 we showed that list-CIHFs (for suitable relations) suffice for applying Fiat-Shamir to the proof system of [GKR15], and to obtain NIZK for $\mathcal{NP}$. In this section we show that these list-CIHFs can be constructed from hardness assumptions, which can serve as alternative assumptions to the ones in Theorems 1.3 and 1.8.

A targeted generator. We will use the following reconstructive targeted PRG from [CT21]. This targeted PRG translates the non-batch-computability (see below) of a function f at input x into pseudorandomness at the same input x. The crucial point is that the trade-off between hardness and randomness is *instance-wise*; that is, the trade-off holds for every *fixed* input x.

The notion of non-batch-computability here means the following. Let $f\colon \{0,1\}^n \to \{0,1\}^k$, and assume each output bit of f can be computed in time T; that is, there is a T-time machine computing the mapping $(x,i) \mapsto f(x)_i$. It is trivial to compute the k-bit output $f(x)$ in time $k \cdot T$, and we will assume that it is impossible to compute $f(x)$ in time $k^\epsilon \cdot T$, for some small $\epsilon > 0$. In fact, we will assume that it is impossible to even approximate $f(x)$ in time $k^\epsilon \cdot T$ (i.e., to print $\tilde{f}(x)$ that agrees with $f(x)$ on $0.99 \cdot k$ coordinates).

Theorem 7.14 ([CT21, Proposition 6.2],[CT23, Theorem 7.1])**.** *For every $\alpha, \beta > 0$ and sufficiently small $\eta = \eta_{\alpha,\beta} > 0$ the following holds. Let $T, k\colon \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $T(n) \geq n$, and let $f\colon \{0,1\}^N \to \{0,1\}^k$ (where $k = k(n)$) such that the mapping of $(x,i) \in \{0,1\}^n \times [k]$ to $f(x)_i$ is computable in time $T(n)$. Then, there is a deterministic algorithm G_f and a probabilistic algorithm* Rec *that for every $x \in \{0,1\}^n$:*

1. **Generator.** *When G_f gets input x, it runs in time $k \cdot T(n) + \mathrm{poly}(k)$ and outputs a list of $\mathrm{poly}(k)$ strings in $\{0,1\}^{k^\eta}$.*
2. **Reconstruction.** *When* Rec *gets input x and oracle access to a $(1/k^\eta)$-distinguisher $D_x\colon \{0,1\}^{k^\eta} \to \{0,1\}$ for the output-list of $G_f(x)$, it runs in time $\tilde{O}(k^{1+\beta}) + k^\beta \cdot T$, makes $\tilde{O}(k^{1+\beta})$ queries to D_x, and with probability at least $1 - 2^{-k^\eta}$ prints a string that agrees with $f(x)$ on at least $1 - \alpha$ of the bits.*

In our applications we will assume that non-batch-computability of f holds when x is sampled from any efficiently samplable distribution, and deduce that the targeted PRG is pseudorandom for inputs coming from any efficiently samplable distribution. As explained in [CT21; CT23], the weaker notion of average-case non-batch-computability over the *uniform distribution* follows from standard assumptions, using the direct product construction of Impagliazzo *et al.* [IJK+10].

Deducing $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$ from non-batch-computability. By combining Theorem 7.14 and Corollary 7.6, we can deduce the conclusion of Theorem 1.3 from an alternative assumption. Specifically, we show that non-batch-computability (with a linear-space oracle) over all efficiently samplable distributions implies that $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$.

Corollary 7.15 (non-batch-computability implies $\mathcal{PSPACE} \subseteq$ cs-$\mathcal{NP}$)**.** *For a sufficiently small $\epsilon > 0$, assume that there is $f\colon \{0,1\}^n \to \{0,1\}^{k=n^\epsilon}$ such that each output bit of f is computable in time $T(n) = \mathrm{poly}(n)$, but the following holds. For every algorithm* Rec *running in time $T \cdot k^{.01}$ and making queries to a linear-space oracle $\mathcal{O}$, and every $2^{O(n)}$-time samplable distribution $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$, and every large enough $n \in \mathbb{N}$, there is at most $2^{-\omega(n)}$ probability over $x \sim \mathbf{x}_n$ that*

$$\Pr[\mathsf{Rec}^{\mathcal{O}}(x)_i = f(x)_i] \geq 0.99\ ,$$

where the probability above is over the random coins of Rec *and over $i \in [k]$.*

Then, there is a $(1/2, 2^{-O(n)})$-targeted HSG running in polynomial time that is pseudorandom for linear space over all $2^{O(n)}$-time samplable distributions, and on inputs of length N outputs $N^{O(\epsilon)}$ strings of length N^δ (where $\delta = \delta_\epsilon > 0$ is a small constant). Consequently, using Corollary 7.6, there is a $\mathcal{PSPACE}$-complete problem decidable in cs-$\mathcal{NTIME}[\mathrm{poly}(n), 2^{O(n)}]$.

Indeed, the constants .01 and 0.99 are arbitrary (corresponding to values of $\alpha = .99 + o(1)$ and $\beta = .01$ in Theorem 7.14) and can be replaced with any pair of constants in $(0,1)$. Also, as explained after Corollary 7.6, the running time $2^{O(n)}$ in the hardness assumption and of the honest prover can be relaxed to 2^{n^ϵ} for any constant $\epsilon > 0$.

Deducing NIZK for $\mathcal{NP}$ from non-batch-computability. In similar fashion, we can deduce the conclusion of Theorem 1.8 from non-batch-computability (with an $\mathcal{NP}$ oracle) over all polynomial-time samplable distributions. Specifically, by combining Theorem 7.14 and Theorem 7.8, we get:

Corollary 7.16 (non-batch-computability implies NIZK for $\mathcal{NP}$)**.** *For a sufficiently small $\epsilon > 0$, assume that there is $f\colon \{0,1\}^n \to \{0,1\}^{k=n^\epsilon}$ such that each output bit of f is computable in time $T(n) = \mathrm{poly}(n)$, but the following holds. For every algorithm* Rec *running in time $T \cdot k^{.01}$ and making queries to a* SAT *oracle $\mathcal{O}$, and every polynomial-time samplable distribution $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$, and every large enough $n \in \mathbb{N}$, there is at most negligible probability over $x \sim \mathbf{x}_n$ that*

$$\Pr[\mathsf{Rec}^{\mathcal{O}}(x)_i = f(x)_i] \geq 0.99\ ,$$

where the probability above is over the random coins of Rec *and over $i \in [k]$.*

Then, there is a $(1/2, n^{-\omega(1)})$-targeted HSG running in polynomial time that is pseudorandom for $\mathcal{DTIME}[n]^{\mathsf{SAT}}$ over all polynomial-time samplable distributions, and on inputs of length N outputs $n^{O(\epsilon)}$ strings of length N^ϵ. If we also assume subexponentially secure one-way functions, then every $\mathcal{NP}$ relation has a NIZK *argument system.*

Acknowledgements

We are very grateful to an anonymous reviewer for pointing out that the function in Assumption 1.2 can be used as-is to obtain zaps and NIWIs.

We thank Alex Lombardi and Vinod Vaikuntanathan for an insightful discussion, and in particular for suggesting the possibility of improving the running time of the verifier in [CT23] to be polynomial. We also thank Ján Pich for pointing out Theorem 1.5 to us, and for suggesting a more elegant proof for it than our original one; in fact, Ján pointed out this potential corollary to us before we obtained our results. We are grateful Yuval Ishai and Mor Weiss for invaluable discussions on zero-knowledge PCPs and beyond. We thank Avi Wigderson for an insightful discussion, and Oded Goldreich for coining the term "computationally sound $\mathcal{NP}$".

This paper started at the "Minimal Complexity Assumptions for Cryptography Workshop" as part of the Meta Complexity program at the Simons Institute. We thank the organizers of the workshop and of the program for inviting us and the Simons Institute for hosting us.

Lijie Chen is supported by a Miller Research Fellowship. Ron Rothblum is funded by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[ACY22] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "Hardness of approximation for stochastic problems via interactive oracle proofs". In: *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*. 2022, Art. No. 24, 16.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3 (1998), pp. 501–555.

[AR23] Noga Amit and Guy N. Rothblum. "Constant-Round Arguments from One-Way Functions". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1537–1544. DOI: 10.1145/3564246.3585244. URL: https://doi.org/10.1145/3564246.3585244.

[Ats06] Albert Atserias. "Distinguishing SAT from polynomial-size circuits, through black-box queries". In: *Proc. 21st Annual IEEE Conference on Computational Complexity (CCC)*. 2006, 8 pp.–95.

[BBH+19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. "On the (In)security of Kilian-Based SNARGs". In: *Proc. 17th Theory of Cryptography Conference (TCC)*. 2019, pp. 522–551.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum disclosure proofs of knowledge". In: vol. 37. 2. 1988, pp. 156–189.

[BF99] Harry Buhrman and Lance Fortnow. "One-Sided Versus Two-Sided Error in Probabilistic Computation". In: *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. 1999, pp. 100–109.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. "Statistical ZAP Arguments". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12107. Lecture Notes in Computer Science. Springer, 2020, pp. 642–667.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. "NIZK from LPN and Trapdoor Hash via Correlation Intractability for Approximable Relations". In: *Proc. 40th Advances in Cryptology - CRYPTO*. 2020, pp. 738–767.

[BKP+23] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. "Batch Proofs are Statistically Hiding". In: *Electron. Colloquium Comput. Complex.* TR23-077 (2023). ECCC: TR23-077. URL: https://eccc.weizmann.ac.il/report/2023/077.

[Blu82] Manuel Blum. "Coin Flipping by Telephone - A Protocol for Solving Impossible Problems". In: *Proc. 24th Computer Society International Conference COMPCON*. 1982, pp. 133–137.

[Blu86] Manuel Blum. "How to prove a theorem so no one else can claim it". In: *Proceedings of the International Congress of Mathematicians*. Vol. 1. Citeseer. 1986, p. 2.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. "Lower bounds for non-black-box zero knowledge". In: *J. Comput. Syst. Sci.* 72.2 (2006), pp. 321–391.

[BM88] László Babai and Shlomo Moran. "Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes". In: *Journal of Computer and System Sciences* 36.2 (1988), pp. 254–276.

[BP15] Nir Bitansky and Omer Paneth. "ZAPs and Non-Interactive Witness Indistinguishability from Indistinguishability Obfuscation". In: *Proc. 12th Theory of Cryptography Conference (TCC)*. 2015, pp. 401–427.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. "Perfect Structure on the Edge of Chaos - Trapdoor Permutations from Indistinguishability Obfuscation". In: *Proc. 13th Theory of Cryptography Conference (TCC)*. 2016, pp. 474–502.

[BTW10] Andrej Bogdanov, Kunal Talwar, and Andrew Wan. "Hard Instances for Satisfiability and Quasi-one-way Functions". In: *Proc. 1st Conference on Innovations in Theoretical Computer Science (ITCS)*. 2010.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 1082–1090.

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. "On the correlation intractability of obfuscated pseudorandom functions". In: *Proc. 19th Theory of Cryptography Conference (TCC)*. 2016, pp. 389–415.

[CCR+18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. "Fiat-Shamir and correlation intractability from strong KDM-secure encryption". In: *Advances in cryptology—EUROCRYPT*. 2018, pp. 91–122.

[CGG+00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "Resettable zero-knowledge (extended abstract)". In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*. Ed. by F. Frances Yao and Eugene M. Luks. ACM, 2000, pp. 235–244.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. "The random oracle methodology, revisited". In: *Journal of the ACM* 51.4 (2004), pp. 557–594.

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. "Correlation Intractability and SNARGs from Sub-exponential DDH". In: *Proc. 43rd Advances in Cryptology - CRYPTO*. 2023, pp. 635–668.

[CHK+19] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. "Finding a Nash equilibrium is no easier than breaking Fiat-Shamir". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 1103–1114.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. "SNARGs for $\mathcal{P}$ from LWE". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 68–79.

[CJJ+23] Geoffroy Couteau, Abhishek Jain, Zhengzhong Jin, and Willy Quach. "A Note on Non-interactive Zero-Knowledge from CDH". In: *Proc. 43rd Advances in Cryptology - CRYPTO*. 2023, pp. 731–764.

[CJS+21] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. "Constructive Separations and Their Consequences". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 646–657.

[CKK+21] Marco Carmosino, Valentine Kabanets, Antonina Kolokolova, and Igor C. Oliveira. "LEARN-Uniform Circuit Lower Bounds and Provability in Bounded Arithmetic". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. 2021, pp. 770–780.

[CKS+21] Geoffroy Couteau, Shuichi Katsumata, Elahe Sadeghi, and Bogdan Ursu. "Statistical ZAPs from Group-Based Assumptions". In: *Proc. 19th Theory of Cryptography Conference (TCC)*. 2021, pp. 466–498.

[CLO24] Lijie Chen, Jiatu Li, and Igor C. Oliveira. "Reverse mathematics of complexity lower bounds". In: *Proc. 65th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2024.

[CN10] Stephen Cook and Phuong Nguyen. *Logical foundations of proof complexity*. Vol. 11. Cambridge University Press Cambridge, 2010.

[CRT22] Lijie Chen, Ron D. Rothblum, and Roei Tell. "Unstructured Hardness to Average-Case Randomness". In: *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022.

[CT21] Lijie Chen and Roei Tell. "Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 125–136.

[CT23] Lijie Chen and Roei Tell. "When Arthur has Neither Random Coins nor Time to Spare: Superfast Derandomization of Proof Systems". In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. 2023.

[CTW23] Lijie Chen, Roei Tell, and R. Ryan Williams. "Derandomization vs Refutation: A Unified Framework for Characterizing Derandomization". In: *Proc. 64th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2023, pp. 1008–1047.

[DFG13] Shlomi Dolev, Nova Fandina, and Dan Gutfreund. "Succinct permanent is *NEXP*-hard with many hard instances". In: *International Conference on Algorithms and Complexity*. Vol. 7878. Lecture Notes in Comput. Sci. 2013, pp. 183–196.

[DFK+92] Cynthia Dwork, Uriel Feige, Joe Kilian, Moni Naor, and Shmuel Safra. "Low Communication 2-Prover Zero-Knowledge Proofs for NP". In: *Proc. 12th Advances in Cryptology - CRYPTO*. 1992, pp. 215–227.

[DN07] Cynthia Dwork and Moni Naor. "Zaps and Their Applications". In: *SIAM J. Comput.* 36.6 (2007), pp. 1513–1543.

[DN93] Cynthia Dwork and Moni Naor. "Pricing via Processing or Combatting Junk Mail". In: *Proc. 13th Advances in cryptology—CRYPTO*. 1993, pp. 139–147.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. "Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract)". In: *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 308–317.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. "Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions". In: *SIAM Journal of Computing* 29.1 (1999), pp. 1–28.

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Advances in cryptology—CRYPTO*. 1986, pp. 186–194.

[FS90] Uriel Feige and Adi Shamir. "Witness Indistinguishable and Witness Hiding Protocols". In: *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*. 1990, pp. 416–426.

[GJJ+20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. "Statistical Zaps and New Oblivious Transfer Protocols". In: *Proc. 39th Advances in Cryptology - EUROCRYPT*. 2020, pp. 668–699.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating computation: interactive proofs for muggles". In: *Journal of the ACM* 62.4 (2015), 27:1–27:64.

[GO94] Oded Goldreich and Yair Oren. "Definitions and Properties of Zero-Knowledge Proof Systems". In: *J. Cryptol.* 7.1 (1994), pp. 1–32.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3.

[Gol11] Oded Goldreich. "Two Comments on Targeted Canonical Derandomizers". In: *Electronic Colloquium on Computational Complexity: ECCC* 18 (2011), p. 47.

[Gol18] Oded Goldreich. "On doubly-efficient interactive proof systems". In: *Foundations and Trends® in Theoretical Computer Science* 13.3 (2018), front matter, 1–89.

[Gol93] Oded Goldreich. "A Uniform-Complexity Treatment of Encryption and Zero-Knowledge". In: *J. Cryptol.* 6.1 (1993), pp. 21–53.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *Proc. 26th Advances in Cryptology - CRYPTO*. 2006, pp. 97–111.

[GS89] Yuri Gurevich and Saharon Shelah. "Nearly linear time". In: *Logic at Botik, Symposium on Logical Foundations of Computer Science*. Lecture Notes in Computer Science. 1989, pp. 108–118.

[GSTS07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "If NP languages are hard on the worst-case, then it is easy to find their hard instances". In: *Computational Complexity* 16.4 (2007), pp. 412–441.

[Gut06] Dan Gutfreund. "Worst-case vs. algorithmic average-case complexity in the polynomial-time hierarchy". In: *Approximation, randomization and combinatorial optimization*. Vol. 4110. Lecture Notes in Comput. Sci. 2006, pp. 386–397.

[GZ11] Oded Goldreich and David Zuckerman. "Another Proof That $\mathcal{BPP} \subseteq \mathcal{PH}$ (and More)". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Ed. by Oded Goldreich. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 40–53.

[Hås87] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal of Computing* 28.4 (1999), pp. 1364–1396.

[HJK+22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. "SNARGs for P from Sub-exponential DDH and QR". In: *Proc. 41st Advances in Cryptology - EUROCRYPT*. 2022, pp. 520–549.

[HL18] Justin Holmgren and Alex Lombardi. "Cryptographic hashing from strong one-way functions". In: *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 850–858.

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. "Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge)". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 750–760.

[HMR08] Shai Halevi, Steven Myers, and Charles Rackoff. "On seed-incompressible functions". In: *Proc. 5th Theory of Cryptography Conference (TCC)*. 2008, pp. 19–36.

[HVW22] Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. "ZK-PCPs from Leakage-Resilient Secret Sharing". In: *J. Cryptol.* 35.4 (2022), p. 23.

[IJK+10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. "Uniform direct product theorems: simplified, optimized, and derandomized". In: *SIAM Journal of Computing* 39.4 (2010), pp. 1637–1665.

[IKO+09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-Knowledge Proofs from Secure Multiparty Computation". In: *SIAM J. Comput.* 39.3 (2009), pp. 1121–1152.

[IW97] Russell Impagliazzo and Avi Wigderson. "P = BPP if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1997, pp. 220–229.

[IWY16] Yuval Ishai, Mor Weiss, and Guang Yang. "Making the Best of a Leaky Situation: Zero-Knowledge PCPs from Leakage-Resilient Circuits". In: *Proc. 13th Theory of Cryptography Conference (TCC)*. 2016, pp. 3–32.

[Jeř05] Emil Jeřábek. "Weak Pigeonhole Principle, and Randomized Computation". Available at https://eccc.weizmann.ac.il/resources/pdf/jerabek.pdf. PhD thesis. Charles University in Prague, 2005.

[Jeř07] Emil Jeřábek. "Approximate counting in bounded arithmetic". In: *The Journal of Symbolic Logic* 72.3 (2007), pp. 959–993.

[JJ21] Abhishek Jain and Zhengzhong Jin. "Non-interactive Zero Knowledge from Sub-exponential DDH". In: *Proc. 40th Advances in Cryptology - EUROCRYPT*. 2021, pp. 3–32.

[JKK+21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. "SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 708–721.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 60–73.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability Obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0". In: *Proc. 41st Advances in Cryptology - EUROCRYPT*. 2022, pp. 670–699.

[Kab01] Valentine Kabanets. "Easiness assumptions and hardness tests: trading time for zero error". In: *Journal of Computer and System Sciences* 63.2 (2001), pp. 236–252.

[Kil92] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proc. 24th Annual ACM Symposium on Theory of Computing (STOC)*. 1992, pp. 723–732.

[KLV23] Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. "SNARGs and PPAD hardness from the decisional Diffie-Hellman assumption". In: *Proc. Advances in cryptology (EUROCRYPT)*. 2023, pp. 470–498.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses". In: *SIAM J. Comput.* 31.5 (2002), pp. 1501–1526.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. "Probabilistically Checkable Proofs with Zero Knowledge". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 496–505.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. "How to delegate computations publicly". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 1115–1124.

[Kra19] Jan Krajíček. *Proof complexity*. Vol. 170. Cambridge University Press, 2019.

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*. Vol. 60. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1995, pp. xiv+343. ISBN: 0-521-45205-8.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. "From obfuscation to the security of Fiat-Shamir for proofs". In: *Advances in cryptology—CRYPTO*. 2017, pp. 224–251.

[Lau83] Clemens Lautemann. "BPP and the polynomial hierarchy". In: *Information Processing Letters* 17.4 (1983), pp. 215–217.

[Les22] Maya Leshkowitz. "Round complexity versus randomness complexity in interactive proofs". In: *Theory of Computing* 18 (2022), Paper No. 13, 65.

[LFK+92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the Association for Computing Machinery* 39.4 (1992), pp. 859–868.

[LO23] Jiatu Li and Igor C. Oliveira. "Unprovability of strong complexity lower bounds in bounded arithmetic". In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. 2023, pp. 1051–1057.

[LVW19] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. "2-Message Publicly Verifiable WI from (Subexponential) LWE". In: *IACR Cryptol. ePrint Arch.* (2019), p. 808.

[LVW20] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. "Statistical ZAPR Arguments from Bilinear Maps". In: *Proc. 39th Advances in Cryptology - EUROCRYPT*. 2020, pp. 620–641.

[LY94] Richard J. Lipton and Neal E. Young. "Simple Strategies for Large Zero-Sum Games with Applications to Complexity Theory". In: *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*. 1994, 734–740.

[Mic00] Silvio Micali. "Computationally sound proofs". In: *SIAM Journal of Computing* 30.4 (2000), pp. 1253–1298.

[MP20] Moritz Müller and Ján Pich. "Feasibly constructive proofs of succinct weak circuit lower bounds". In: *Annals of Pure and Applied Logic* 171.2 (2020), pp. 102735, 45.

[MS23a] Dieter van Melkebeek and Nicollas Sdroievski. "Instance-wise hardness versus randomness tradeoffs for Arthur-Merlin protocols". In: *Proc. 38th Annual IEEE Conference on Computational Complexity (CCC)*. 2023, Art. No. 17, 36.

[MS23b] Dieter van Melkebeek and Nicollas Sdroievski. "Leakage resilience, targeted pseudorandom generators, and mild derandomization of Arthur-Merlin protocols". In: *Proc. 43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2023, Art. No. 29, 22.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. "Derandomizing Arthur-Merlin games using hitting sets". In: *Computational Complexity* 14.3 (2005), pp. 256–279.

[Nao03] Moni Naor. "On Cryptographic Assumptions and Challenges". In: *Proc. 23rd Advances in cryptology—CRYPTO*. Ed. by Dan Boneh. Springer, 2003.

[Nao91] Moni Naor. "Bit Commitment Using Pseudorandomness". In: *J. Cryptol.* 4.2 (1991), pp. 151–158.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. "Hardness Magnification for Natural Problems". In: *Electronic Colloquium on Computational Complexity: ECCC* 25 (2018), p. 139.

[OV07] Boaz Barakand Shien Jin Ong and Salil P. Vadhan. "Derandomization in Cryptography". In: *SIAM J. Comput.* 37.2 (2007), pp. 380–400.

[Pic15a] Ján Pich. "Circuit lower bounds in bounded arithmetics". In: *Annals of Pure and Applied Logic* 166.1 (2015), pp. 29–45.

[Pic15b] Ján Pich. "Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic". In: *Logical Methods in Computer Science* 11.2 (2015), 2:8, 38.

[PS19] Chris Peikert and Sina Shiehian. "Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors". In: *Advances in Cryptology - CRYPTO*. 2019, pp. 89–114.

[PS21] Ján Pich and Rahul Santhanam. "Strong co-nondeterministic lower bounds for NP cannot be proved feasibly". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 223–233.

[Rab79] Michael O Rabin. "Digitalized signatures and public-key functions as intractable as factorization". In: (1979).

[Raz85] A. A. Razborov. "Lower bounds on the monotone complexity of some Boolean functions". In: *Doklady Akademii Nauk SSSR* 281.4 (1985), pp. 798–801.

[Raz87] Alexander A. Razborov. "Lower bounds on the size of constant-depth networks over a complete basis with logical addition". In: *Mathematical Notes of the Academy of Science of the USSR* 41.4 (1987), pp. 333–338.

[Raz95] Alexander A. Razborov. "Bounded arithmetic and lower bounds in Boolean complexity". In: *Feasible mathematics, II (Ithaca, NY, 1992)*. Vol. 13. Progr. Comput. Sci. Appl. Logic. 1995, pp. 344–386.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-round interactive proofs for delegating computation". In: *SIAM Journal of Computing* 50.3 (2021), STOC16–255–STOC16–340.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126.

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

[Sha92] Adi Shamir. "IP = PSPACE". In: *Journal of the ACM* 39.4 (1992), pp. 869–877.

[Sip83] Michael Sipser. "A complexity theoretic approach to randomness". In: *Proc. 15th Annual ACM Symposium on Theory of Computing (STOC)*. 1983, pp. 330–335.

[Smo87] Roman Smolensky. "Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity". In: *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*. 1987, pp. 77–82.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom generators without the XOR lemma". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. "Simple extractors for all min-entropies and a new pseudorandom generator". In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[SU07] Ronen Shaltiel and Christopher Umans. "Low-end uniform hardness vs. randomness tradeoffs for AM". In: *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 2007, pp. 430–439.

[TV07] Luca Trevisan and Salil P. Vadhan. "Pseudorandomness and Average-Case Complexity Via Uniform Reductions". In: *Computational Complexity* 16.4 (2007), pp. 331–364.

[Ver13] Nikolay Vereshchagin. "Improving on Gutfreund, Shaltiel, and Ta-Shma's paper "If NP languages are hard on the worst-case, then it is easy to find their hard instances"". In: *Computer science—theory and applications*. Vol. 7913. Lecture Notes in Comput. Sci. 2013, pp. 203–211.

[Wei22] Mor Weiss. "Shielding Probabilistically Checkable Proofs: Zero-Knowledge PCPs from Leakage Resilience". In: *Entropy* 24.7 (2022), p. 970.

A Derandomizing the [HVW22] zkPCP

The zkPCP of [HVW22] that we use is based on the "MPC in the head" technique of [IKO+09]. For a given $\mathcal{NP}$ relation R and parameter $q_{\max}$, the PCP is constructed as follows. Let $t = O(q_{\max})$ and $\ell = 2t$, the prover runs a *maliciously secure* ℓ-player MPC protocol, with perfect security, wrt the functionality $f(x, w_1, \ldots, x_\ell) = R(x, \oplus_i w_i)$, where the input to the i-th player is (x, w_i) and the output is 1 if and only if $(x, \oplus_i w_i) \in R$. The PCP proof-string is constructed by having each PCP symbol include the entire view of one of the ℓ players.

The PCP verifier chooses at random a set $S \subseteq [\ell]$ of size $q = \tilde{O}(\sqrt{\ell})$ of the players and checks that their views are (1) consistent with each other (i.e., the outgoing message from any player $i \in S$ to player $j \in S$ is consistent with message received by player j, and (2) that all the of selected players output 1.

The analysis of [HVW22] chooses the set S entirely at random which requires randomness complexity $\tilde{O}(\sqrt{q_{\max}})$, which is too large for our purposes. Next, we describe how to reduce the randomness complexity to $O(\log \ell) = O(\log(q_{\max}))$. We remark that in contrast to [HVW22] we only aim to achieve a constant soundness error.

Our derandomization of the PCP will only use uses a subset of the query sets that the verifier of [HVW22] makes, and the prover remains unchanged. This means that we automatically inherit the (perfect) completeness and zero-knowledge from [HVW22].

For soundness, the key point in the analysis of [HVW22] (following [IKO+09]) is to show that with all but negligible probability, when S is chosen at random, then either a player who's view is not corrupted was selected (in which case the verifier will reject due to the malicious security of the MPC protocol), or alternatively, the views of two of the selected players are inconsistent with each other.

Establishing soundness boils therefore to the following combinatorial problem - given a graph[57] G on ℓ vertices

- If the graph has a small vertex cover B then with high probability $S \cap \bar{B} \neq \emptyset$ (i.e., a vertex *outside* the vertex cover is selected).
- If the graph does not have a small vertex cover, then with high probability, the two endpoints of one of its edges are selected.

We show that the same properties hold if we sample the vertices from a 4-wise independent distribution. In more detail, we sample the set S as follows. Let $X = (X_v)_{v \in V}$ be a 4-wise independent distribution where each X_v is a Bernoulli random variable with mean q/t, where $q = \Theta(\sqrt{t})$. We let $S = \{v : X_v = 1\}$.[58]

Our analysis uses the following fact that follows immediately from Chebyshev's inqequality.

[57] This graph, called the *inconsistency graph* [IKO+09], has ℓ vertices, one per player, and an edge (u, v) whenever the views of u and v are inconsistent, see [IKO+09; HVW22] for details.

[58] To have a worst-case bound on the size of S, in case $|S| > 2\ell q/t$ the verifier immediately accepts. By Chebyshev's inequality this increases the soundness error by at most $O(1/q^2)$.

Fact A.1. *Let $X_1, \ldots, X_n$ be pairwise independent Bernoulli random variables with the same mean $p \in (0,1]$. Then,*

$$\Pr[X_1 = \cdots = X_n = 0] \leq \frac{1}{np}.$$

We consider now the two cases:

- Case I: the graph contains a vertex cover B of size at most t. Observe that $(X_v)_{v \notin B}$ is a sequence of at least $\ell - t \geq t$ pairwise independent Bernoulli random variables with mean q/t. Thus, by Fact A.1, it holds that they are all 0 (i.e., none of them are selected) with probability at most $1/q$.
- Case II: the graph does not contain a vertex cover of size at most t. Then, following [IKO+09], the graph must contain a matching M of size at least $t/2$. For every edge $e = (u,v) \in M$, let $Y_e = X_u \cdot X_v$. That is, Y_e is an indicator for the event that $u, v \in S$. Observe that since M is a matching, and the (X_v) variables are 4-wise independent, the variables $(Y_e)_{e \in M}$ are pairwise independent. Thus, $(Y_e)_{e \in M}$ is a sequence of at least $t/2$ pairwise independent Bernoulli random variables with mean $\mathbb{E}[Y_e] = \mathbb{E}[X_u] \cdot \mathbb{E}[X_v] = (q/t)^2$. Hence, by Fact A.1 the probability that $Y = 0$ is at most $\frac{2t}{q^2}$.

Taking $q = \Theta(\sqrt{t})$, we get that at least one of the above events happens with constant probability.

Lastly, we remark that the above 4-wise independent sequence can be generated using a 4-wise independent hash function from $[\ell]$ to $[t/q]$, and selecting all inputs that are mapped, say, to 0.

B The FLS Trick for Uniform Adversaries: Proof of Lemma 3.7

Let R be an $\mathcal{NP}$ relation and let G be a length-doubling cryptographic pseudorandom generator (such exists since we assumed one-way functions [HIL+99]). Let $R' = \left\{ ((x,r),w) : r = 2|x| \text{ and } \Big(((x,w) \in R) \text{ or } (G(w) = r) \Big) \right\}$. That is, the relation R' consists of all inputs (x,r) such that either x is YES instance of R *or* r is in the image of the PRG. Note that $R' \in \mathcal{NP}$ and so by our hypothesis it has a zap.

For a given input length n, the NIZK CRS consists of a CRS crs for a Zap for L' (on input of length $n + 2n$), as well as a uniformly random string $r \in \{0,1\}^{2n}$. Given input x, the prover P and verifier V run the zap on the statement $((x,r),w)) \in R'$, where w is the $\mathcal{NP}$ witness given to the prover.

Completeness follows from the completeness of the zap, while observing that $(x,w) \in R$ and and so $((x,r),w) \in R'$.

For soundness, let P^* be a ppt cheating prover. Suppose that P^*, given input (crs, r) outputs (x, α) such that $V((crs,r),x,\alpha) = 1$ and $x \in \{0,1\}^n \backslash L(R)$, with probability at least ϵ. We use P^* to violate the soundness of the zap as follows - given as input a CRS crs the prover chooses $r \in \{0,1\}^{2n}$ uniformly at random and then uses $P^*(crs, r)$ to obtain (x, α) and supplies α as the proof, relative to the input (x,r). Observe that this proof convinces the zap verifier with the same probability ϵ, and that will all but 2^{-n} probability, it holds that $r \notin Im(G)$ and so $(x,r) \notin L(R')$. Overall we obtain a violation of the soundness of the zap with advantage $\epsilon - 2^{-n}$.

To show the zero-knowledge property, we construct a simulator (S_1, S_2) as follows. The first algorithm S_1, on input 1^n, first generates $s \in \{0,1\}^n$ and crs uniformly at random. The CRS is then set to be (crs, r), where $r = G(s)$, and the auxiliary input (aka trapdoor) is $\tau = (crs, s)$. The second algorithm S_2, given (x, τ), interprets τ as (crs, s), sets $r = G(s)$, and outputs the NIZK proof $\alpha \leftarrow P_{R'}(crs, (x, r), s)$ (indeed s is used as the witness for the fact that $(x, r) \in L'$).

Assume toward a contradiction that (S_1, S_2) do not satisfy the zero-knowledge property. Thus, there exist a pair of ppt algorithms (A_1, A_2) such that

$$\left| \Pr_{\substack{crs \\ r \in \{0,1\}^{2n} \\ (x,w,z) \leftarrow A_1(crs,r) \\ \alpha \leftarrow P_{R'}(crs,(x,r),w)}} \left[\begin{array}{c} A_2((crs,r),x,\alpha,z) = 1 \\ \wedge \\ (x,w) \in R \end{array} \right] - \Pr_{\substack{crs \\ s \in \{0,1\}^n \\ (x,w,z) \leftarrow A_1(crs,G(s)) \\ \alpha \leftarrow P_{R'}(crs,(x,G(s)),s)}} \left[\begin{array}{c} A_2((crs,G(s)),x,\alpha,z) = 1 \\ \wedge \\ (x,w) \in R \end{array} \right] \right|$$

is non-negligible. By the security of the PRG, since $P_{R'}$ and (A_1, A_2) are ppt, we have that:

$$\left| \Pr_{\substack{crs \\ r \in \{0,1\}^{2n} \\ (x,w,z) \leftarrow A_1(crs,r) \\ \alpha \leftarrow P_{R'}(crs,(x,r),w)}} \left[\begin{array}{c} A_2((crs,r),x,\alpha,z) = 1 \\ \wedge \\ (x,w) \in R \end{array} \right] - \Pr_{\substack{crs \\ s \in \{0,1\}^n \\ (x,w,z) \leftarrow A_1(crs,G(s)) \\ \alpha \leftarrow P_{R'}(crs,(x,G(s)),w)}} \left[\begin{array}{c} A_2((crs,G(s)),x,\alpha,z) = 1 \\ \wedge \\ (x,w) \in R \end{array} \right] \right|$$

is negligible. By the triangle inequality, this implies that:

$$\left| \Pr_{\substack{crs \\ s \in \{0,1\}^n \\ (x,w,z) \leftarrow A_1(crs,G(s)) \\ \alpha \leftarrow P_{R'}(crs,(x,G(s)),w)}} \left[\begin{array}{c} A_2((crs,G(s)),x,\alpha,z) = 1 \\ \wedge \\ ((x,G(s)),w) \in R' \\ \wedge \\ ((x,G(s)),s) \in R' \end{array} \right] - \Pr_{\substack{crs \\ s \in \{0,1\}^n \\ (x,w,z) \leftarrow A_1(crs,G(s)) \\ \alpha \leftarrow P_{R'}(crs,(x,G(s)),s))}} \left[\begin{array}{c} A_2((crs,G(s)),x,\alpha,z) = 1 \\ \wedge \\ ((x,G(s)),w) \in R' \\ \wedge \\ ((x,G(s)),s) \in R' \end{array} \right] \right|,$$

is non-negligible, which contradicts the witness indistinguishibilty property.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il