Non-Uniformly Sound Certificates with Applications to Concurrent Zero-Knowledge

Cody Freitag*, Ilan Komargodski*, and Rafael Pass*

Cornell Tech, New York, NY 10044, USA
cfreitag@cs.cornell.edu
komargodski@cornell.edu
rafael@cs.cornell.edu

Abstract. We introduce the notion of non-uniformly sound certificates: succinct single-message (unidirectional) argument systems that satisfy a "best-possible security" against non-uniform polynomial-time attackers. In particular, no polynomial-time attacker with s bits of non-uniform advice can find significantly more than s accepting proofs for false statements. Our first result is a construction of non-uniformly sound certificates for all **NP** in the random oracle model, where the attacker's advice can depend arbitrarily on the random oracle.
We next show that the existence of non-uniformly sound certificates for **P** (and collision resistant hash functions) yields a *public-coin constant-round* fully concurrent zero-knowledge argument for **NP**.

1 Introduction

We consider the following compression task for a language L in **NP**. An efficient prover holds an input x and a witness w to the fact that $x \in L$ and wishes to send to a verifier a "short" *certificate* π testifying to the validity of x. The length of the certificate should be independent of the length of the statement x, the witness w, and the time needed to verify that w is a witness for x. In other words, the same proof length n can be used for every **NP** language L and all statements x with arbitrary polynomial length in n. The verifier should be able to determine whether π is a valid certificate for a statement x in time polynomial in $|x|, |\pi|$. We refer to this "witness compression" task as a *succinct* certificate system, or simply a certificate system.

By the result of Goldreich and Håstad [29], certificate systems for **NP** with statistical soundness (i.e., soundness against unbounded provers) imply that **NP** can be decided in subexponential time, and thus are unlikely to exist. Even for **P**, certificate systems imply that non-determinism can speed up arbitrary polynomial-time computation, contradicting widely believed derandomization assumptions (e.g., Barak et al. [4,5]). Thus, we are interested in certificate systems with *computational* soundness, where soundness holds against efficient attackers. Note that the notion of SNARGs (succinct non-interactive arguments)

* Supported in part by NSF Award CNS-1561209, NSF Award SATC-1617676, NSF Award SATC-1704788, and AFOSR Award FA9550-15-1-0262.

[45,26] satisfies our efficiency and soundness requirements, but are actually not "unidirectionally" non-interactive: rather, the verifier (or some other trusted entity) must first generate and send a public parameter (which can be reused over multiple proofs) to the prover which the prover can use to produce its proof. Rather, in a certificate system, there is no *a priori* agreed-upon public parameters and no communication from the verifier to the prover.

The problem in such a fully non-interactive setting is that the standard notion of computational soundness (against non-uniform polynomial-time attackers) trivially collapses down to the notion of statistical soundness: if a cheating certificate π exists for some statement x (violating statistical soundness), then an efficient non-uniform attacker can simply get the pair (x, π) as non-uniform advice and consequently break computational soundness. Note that this attack is no longer possible for SNARGs when the non-uniform advice cannot be chosen as a function of the public parameters.

On best-possible security. One approach to overcome the above problem is to settle for soundness against only *uniform* polynomial-time attackers. Certificate systems satisfying such a uniform notion of soundness were considered for $\mathbf{P}$ under the name $\mathbf{P}$-certificates by Chung, Lin, and Pass [14]. They also observed that Micali's CS-proofs [45] satisfy uniform soundness in the random oracle model. But, there is a reason security against non-uniform attackers has become the standard notion of security in the cryptographic literature: it captures the natural idea that an adversary may have been designed to attack specific instances, guaranteeing security against an expensive preprocessing stage or any unknown future attacks.

Of course, s bits of non-uniform advice can be used to encode roughly s accepting proofs for false statements. But can they be used to encode much more? This motivates the following definition of *best-possible soundness* in the language of computational Kolmogorov complexity: the computational Kolmogorov complexity of K accepting proofs of different false statements cannot be significantly smaller than K. In other words, false certificates are "incompressible."

Equivalently, in a more complexity-theoretic language, we consider a notion of "multi-statement" soundness which requires that no non-uniform polynomial-time adversary having non-uniform advice of length s can produce accepting certificates for significantly more than s false statement. For a class of problems $\mathcal{C}$, we call argument systems that satisfy such a notion as *non-uniformly sound certificates for* $\mathcal{C}$, or $\mathsf{nuCerts}$ for $\mathcal{C}$ in short.

On the existence of $\mathsf{nuCerts}$. We initiate the study of $\mathsf{nuCerts}$. We first note that whereas [14] observed that Micali's CS-proofs are uniformly sound in the random-oracle model, they do *not* satisfy multi-statement soundness. In fact, an efficient cheating prover with just polynomially-many bits of advice about the random oracle can produce proofs for *exponentially* many false statements! The same happens for other SNARG constructions which use only a random string as a public parameter (and for SNARGs that use a non-random public parameter, it is not clear how to turn them into certificates).

Our first main result is a construction of nuCerts in the random oracle model. Formally, we prove that our construction satisfies multi-statement soundness in the auxiliary-input random oracle model. The auxiliary-input random oracle model (AI-ROM), introduced by Unruh [55], captures preprocessing attacks, where the non-uniform advice string given to the attacker can depend arbitrarily on the random oracle.

Theorem 1 (Informal). *There exist* nuCerts *for* ***NP*** *in the auxiliary-input random oracle model.*

At a very high level, we present a construction which mimics Micali's construction of CS-proofs. Micali's construction works as follows: (1) start off with an efficient multi-round argument systems for **NP** and next (2) collapse the rounds to make it non-interactive using the Fiat-Shamir heuristic [22] (i.e., use the random oracle to generate verifier messages for the interactive protocol). Whereas the Fiat-Shamir heuristic indeed leads to a sound round reduction for uniform attackers in the random oracle model, in general it does not for non-uniform ones (where a non-uniform attacker can perform unbounded preprocessing on the random oracle). Indeed, as mentioned above, Micali's CS-proofs do not satisfy multi-statement soundness w.r.t. non-uniform attackers. As a warm-up, we show that if the underlying interactive proofs system has 3 rounds and has *statistical soundness*, then Fiat-Shamir with a minor modification in fact works. Unfortunately, this modification of Fiat-Shamir does not suffice for making *any* argument non-interactive. Rather, we present a different variant of Kilian's efficient arguments for **NP** and prove that for this *particular* argument system, the Fiat-Shamir paradigm does lead to a multi-statement sound non-interactive proof (although it does not work for Kilian's original argument). Our proof relies on a quite interesting compression argument which we believe may be of independent interest.

Application: public-coin constant-round concurrent zero-knowledge. Given a language $L \in \mathbf{NP}$ and an instance x, zero-knowledge (ZK) proofs [30] allow (paradoxically) for a prover to convince a verifier of the validity of a mathematical statement $x \in L$, while providing zero additional knowledge to the verifier. This is formalized by requiring that the view of every efficient adversarial verifier interacting with the honest prover be *simulated* by an efficient machine called the simulator. *Concurrent* ZK models the (realistic) asynchronous and concurrent setting, where a single adversary can "attack" the prover by acting as a verifier in many *concurrent* executions. Starting with the original work of Dwork, Naor, and Sahai [19], there has been a long line of constructions of concurrent ZK protocols. These include constructions with black-box simulation (e.g., [54,40,53]) nearly matching the almost logarithmic-round lower bound of [11] (by [51], these protocols are inherently private-coin), constructions based on different setup assumptions (e.g., [20,17,10,28,50,32]) and constructions in alternative less standard models (e.g., super-polynomial-time simulation [48,52] or based on knowledge assumptions [33]).

Using nuCerts for **P**, we give a public-coin, constant-round, (fully) concurrent ZK argument for **NP**. Public-coin protocols are ones where the verifier's messages are simply random coin tosses. This is a natural and appealing property of a protocol, which is useful in various applications such as public verifiability, leakage resilience [24], constructing resettably-sound protocols [3,51], and many more. The security of our construction relies on the following assumptions: the existence of a nuCert for **P** and a family of collision resistant hash functions (both with slightly super-polynomial security).

Theorem 2 (Informal). *Assume the existence of families of collision-resistant hash functions, and the existence of a* nuCert *for* ***P*** *(both with slightly super-polynomial security). Then, there exists a public-coin constant-round concurrent ZK argument for* ***NP****.*

Barak's breakthrough work [1] gives a public-coin constant-round protocol but with *bounded* concurrency. Canetti, Lin, and Paneth [12] achieve full concurrency but require $O(\log^{1+\epsilon} n)$ rounds in the global hash model, and similarly Goyal [31] requires $O(n^{\epsilon})$ rounds based on more standard assumptions (where in both n is the security parameter and $\epsilon > 0$ is an arbitrary small constant). Chung, Lin, and Pass [14] achieve constant rounds assuming **P**-certificates but only with a less standard notion of soundness (*uniform* soundness).

We stress that the construction from Theorem 2 is in the standard model assuming the existence of collision resistant hash functions and nuCerts for **P**. Using Theorem 1, we can instantiate nuCerts for **P** using a (non-programmable) random oracle and get a public-coin constant-round concurrent ZK argument for **NP**.

Corollary 1. *In the auxiliary-input random oracle model, there exists a public-coin constant-round concurrent zero-knowledge argument for* ***NP****.*

Note that our protocol comes with an explicit ZK simulator. For the instantiation in the random oracle model, only soundness relies on the random oracle while ZK holds with any concrete function. Prior to this work, there were no public-coin constant-round constructions that provably satisfy concurrent ZK and even heuristically satisfy non-uniform soundness.

Lastly, even ignoring the public-coin aspect of our protocol, our result is meaningful: The only previously known (private-coin) constant-round fully concurrent ZK protocols rely on obfuscation-type assumptions (Chung, Lin, and Pass [15] rely on indistinguishability obfuscation and Pandey, Prabhakaran and Sahai [47] rely on differing-input obfuscation). Based on the protocol of Chung et al. [15], Chongchitmate, Ostrovsky, and Visconti [13] show a transformation to achieve simultaneously resettable ZK [3,10] in constant rounds based on the same assumptions (including indistinguishability obfuscation). Since our protocol is public-coin, we immediately get a constant-round simultaneously-resettable ZK protocol using known transformations [3,51,18] based on nuCerts for **P** (and hence also in the auxiliary-input random oracle model).

Paper organization. In Section 2, we give an overview of our main techniques. In Section 3, we provide preliminary definitions and standard notation. In Section 4, we formally define nuCerts, and in Section 5, we present our candidate nuCert construction. Lastly, in Section 6, we show how to use nuCerts for $\mathbf{P}$ to get a public-coin constant-round concurrent zero-knowledge protocol.

1.1 Related Work

The idea to define the best-possible security for setup-free non-interactive primitives is inspired by the work of Bitansky, Kalai, and Paneth [7] that considered keyless multi-collision resistant hash functions. These are compressing functions where it is assumed that no efficient adversary with s bits of non-uniform advice can find significantly more than s values all of which collide relative to a fixed hash function. They used such functions to shave off one round of communication in various zero-knowledge protocols. Multi-collision resistant hashing [41,6,7,42,43] was also studied in the keyed setting as a relaxation of plain collision resistance.

In a recent work, Bitansky and Lin [8] considered one-message zero-knowledge arguments, where the soundness guarantee is that the number of false statements that an efficient non-uniform adversary can convince the verifier to accept is not much larger than the size of its non-uniform advice. They constructed such zero-knowledge arguments based on keyless collision resistant hash functions. Note that their construction is not succinct and thus cannot be used in place of our non-uniform certificates.

2 Technical Overview

In this section we provide a high-level overview of our constructions. We start with the non-uniformly sound certificates and then present our concurrent zero-knowledge protocol.

2.1 Non-uniformly Sound Certificates

Let us start with a more elaborate description of Micali's CS-proofs [45] and why they give only *uniformly* sound certificates [14]. Micali's protocol is obtained by applying the Fiat-Shamir heuristic [22] to Kilian's 4-round argument system [38]. In Kilian's protocol, the verifier sends a description of a collision resistant hash function to the prover. Using this hash function, the prover computes a Merkle hash a of a PCP proof for the statement q and sends it back to the verifier. The verifier then sends b, defining a set of PCP challenge queries, and the prover replies with the authentication paths c in the Merkle tree for all openings in the PCP proof specified by b. Micali's protocol collapses Kilian's protocol to one message by using a random oracle to compute the Merkle tree with root a, then uses the random oracle on a to derive the challenge b specifying openings with

authentication paths c. The final proof $\pi = (a, b, c)$ is sent to the verifier to be checked.

One way to argue uniform soundness of Micali's non-interactive protocol is to reduce the security to Kilian's interactive protocol (using the same random oracle for the Merkle tree). Basically, we receive a random challenge b from the interactive protocol and then simulate the non-interactive cheating prover. When the cheating prover queries the random oracle for the value b', we respond instead with the challenge b. (We identify the query for b' by just guessing, which is okay since there are polynomially-many queries.) Since the adversary is uniform, it cannot distinguish b from b', so with noticeable probability, it will output accepting authentication paths, which we forward to the interactive verifier.

The above proof fails to go through when the adversary has non-uniform advice depending on the random oracle. The query for b' is not necessarily random, so the adversary may be able to distinguish b from b' and as a result will fail to output accepting authentication paths. Concretely, a non-uniform attacker for the non-interactive protocol can have hardwired a triple (a, b, c) which causes the verifier to accept for a fixed random oracle. In this case, we cannot change the value of b to make the above reduction go through. But, could it be the case that if the adversary needs to come up with *many* accepting proofs for false statements, at least one will have a random b value? The answer is "no." When the adversary has unbounded preprocessing time, there are many more cheating strategies. For example, the adversary may have hardcoded collisions relative to the random oracle that allow him to explain the root value of the Merkle tree of the statement in exponentially many ways.

Multi-statement sound Fiat-Shamir. A natural first question is to understand how to modify the Fiat-Shamir heuristic to get multi-statement soundness. As a warm-up, let us start with a 3-message, public-coin, succinct *proof* system. Can we make it non-interactive and multi-statement sound? The security guarantee of this protocol is that for any false instance and every message a from the prover, there is a small set of b values that will allow the prover to come up with a c message which causes the verifier to accept. A first attempt to make it non-interactive is to use the random oracle to derive the b value given the a value, $b = \mathcal{O}(a)$. This is completely insecure as it could be the case that there is a b message which always causes the verifier to accept, so all the cheating prover has to do is to find one a that is mapped to b (which it can do in the preprocessing stage). The natural fix is to index the random oracle $\mathcal{O}$ with the statement q, $\mathcal{O}(q, \cdot)$. Then, intuitively, if he wishes to cheat on many statements by sending the bad b message, it needs to find many statements q' with corresponding a' values such that for all pairs, it holds that $\mathcal{O}(q', a') = b$.

Regarding security, we need to prove that our construction has defended against *all possible attacks* in the auxiliary-input random oracle model. The first proof approach to consider in this model is using the bit-fixing random oracle model of Unruh [55] (see also Coretti et al. [16]). At a high level, this approach says that any result in the "uniform" random oracle model can be translated

into a hybrid model where some of the locations of the random oracle have been fixed but the rest are lazily sampled. Then, as long as we can show the adversary will likely query outside the set of fixed points, we can argue soundness just as in the uniform setting. This elegant model unfortunately does not work in the fully non-interactive setting where there is no high entropy setup *independent* of the adversary's advice. Thus, the adversary may only query fixed points for which it has encoded information.

Our proof of security is done via a compression argument, à la Gennaro and Trevisan [25]. We show that if an adversary with only s bits of non-uniform advice can come up with significantly more than s accepting proofs for false statements, then we can use the adversary to *compress* the random oracle. The idea in the proof, at a very high level, is to carefully map all possible ways the adversary can cheat to first encode the random oracle $\mathcal{O}$ in a way that we can still answer all of the adversary's queries. Then, we can run the adversary to uniquely reconstruct $\mathcal{O}$. If the encoding has been compressed more than the adversary's advice, then the adversary can succeed only on a small fraction of random oracles. In the compression argument, we crucially use the fact that the protocol is statistically sound and that we can uniquely extract the statement corresponding to each query in order to enumerate the small set of accepting b values (which is limited by soundness).

Caveat 1: While it may seem like we are done, there is a significant caveat in the above scheme. The issue is that we cannot use the statement q itself to index the random oracle $\mathcal{O}$ since q has no *a priori* size bound. The most we could hope for is to use a *short commitment* to the statement as an index to the random oracle. One could try to use a Merkle tree to implement this approach, i.e., index $\mathcal{O}$ with $\tilde{q} = \mathsf{MerHash}^{\mathcal{O}(\cdot)}(q)$, but it is again completely insecure – a cheating prover can simply encode collisions and use a mix-and-match attack to find exponentially different statements with the same hash value.

Solution 1: A similar issue came up in recent works on domain extension of multi-collision resistant hash functions [42,7]. In both works, they propose to encode the input using a specific code before hashing via a Merkle tree. Following [42], we use *list-recoverable codes* (LRC) to encode each statement before applying a Merkle tree to hash the statement to a short value. That is, for a statement q, we compute $\tilde{q} = \mathsf{MerHash}^{\mathcal{O}(\cdot)}(\mathsf{LRC}(q))$. The LRC guarantees that if the adversary can open each leaf in the Merkle tree to only a polynomial number of values, then there are only a polynomial (rather than exponential) number of possible statements with valid codewords. It follows that the prover can cheat on only a bounded number of hash values and can find only a bounded number of statements per hash value.

Caveat 2: Recall that our compression argument requires us to know how to map each oracle query for b back to the statement q. However, the solution to the previous problem, was to index the queries only with a short commitment to the statement $b = \mathcal{O}(\tilde{q}, a)$. So, we need a way to map this commitment $\tilde{q}$ to a unique statement q (in the proof).

Solution 2: We append to the proof a Merkle hash of the statement using the random oracle $\mathcal{O}(\tilde{q}, \cdot)$ indexed by $\tilde{q}$: $\hat{q} = \mathsf{MerHash}^{\mathcal{O}(\tilde{q},\cdot)}(q)$. Additionally, we derive b by indexing $\mathcal{O}$ with both $\tilde{q}$ and $\hat{q}$: $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$. Then, in our compression argument, we try to use the adversary's queries to $\mathcal{O}(\tilde{q}, \cdot)$ (which it must make to compute $\hat{q}$) to reconstruct the statement q and be able to compress the possible b values by statistical soundness. In doing so, we will compress the random oracle in one of three ways. Either:

1. there is a unique way to reconstruct q, so we can use statistical soundness to compress the oracle query for $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$;
2. there is more than one way to reconstruct q, so the adversary must have found a collision in $\mathcal{O}(\tilde{q}, \cdot)$; or
3. there is no way to reconstruct q, so the adversary must "know" the preimage of some point in $\mathcal{O}(\tilde{q}, \cdot)$.

In summary, the variant of Fiat-Shamir we introduce is to first encode the statement q with a good list-recoverable code $\mathsf{LRC}(q)$ and hash it down to get $\tilde{q} = \mathsf{MerHash}^{\mathcal{O}(\cdot)}(\mathsf{LRC}(q))$. Then, hash down the statement using a random oracle that is indexed by $\tilde{q}$ to get $\hat{q} = \mathsf{MerHash}^{\mathcal{O}(\tilde{q},\cdot)}(q)$. Then, compute $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$ with a random oracle indexed by $\tilde{q}$ and $\hat{q}$.

Multi-statement soundness for arguments. While the above approach works for proof systems, it does not hold generically for arguments (by the same reason that the original Fiat-Shamir heuristic fails on Barak's protocol [1]). However, we show that it does work for Kilian's protocol [38] relying on some specific properties of this protocol.

Our main idea is to leverage the soundness of the underlying PCP proof system. By PCP soundness, for any statement q and any fixed PCP proof string $\Pi = \mathsf{PCP}(q)$, the number of accepting b values must be small. The main technical difficulty is that to use PCP soundness, we need to make sure the adversary is bound to a single PCP proof Π, which we need to know to determine the accepting b values. We use the same compression idea as before in order to extract the statement. Namely, we compute $a = \mathsf{MerHash}^{\mathcal{O}(\tilde{q},\cdot)}(\Pi)$. Then, in the compression argument, either (1) we can uniquely extract a PCP proof string Π from the prover's queries and can compress the query for $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$ by PCP soundness; (2) we reconstruct more than one PCP proof string, so the adversary must know a collision in $\mathcal{O}(\tilde{q}, \cdot)$; or (3) we cannot reconstruct a valid PCP proof, so the adversary must know some preimage for $\mathcal{O}(\tilde{q}, \cdot)$. We show that this covers all possibilities, and for each commitment value $\tilde{q}$ that the adversary cheats on, we will compress the random oracle at a new point.

In summary, our construction consists of a prover and verifier with the following strategies. The prover's strategy given a statement q is:

- Encode the statement with a good list-recoverable code $\mathsf{LRC}(q)$ and hash it down to get $\tilde{q} = \mathsf{MerHash}^{\mathcal{O}(\cdot)}(\mathsf{LRC}(q))$.
- Hash down the statement using a random oracle that is indexed by $\tilde{q}$ to get $\hat{q} = \mathsf{MerHash}^{\mathcal{O}(\tilde{q},\cdot)}(q)$.

– Compute a commitment to a PCP of q using a random oracle indexed by $\tilde{q}$ and $\hat{q}$: $a = \mathsf{MerHash}^{\mathcal{O}(\tilde{q},\hat{q},\cdot)}(\mathsf{PCP}(q))$.
– Compute $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$ with a random oracle indexed by $\tilde{q}$ and $\hat{q}$.
– Let c be the authentication paths corresponding to the indices given by b.
– **Output:** The certificate is $\pi = (a, b, c)$.

The verifier on input $\pi = (a, b, c)$ and q first computes $\tilde{q}$ and $\hat{q}$ in the same way as the prover (this is allowed since the LRC is efficient and the verifier can run in time polynomial in $|q|$). Then it checks that $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$ and checks the validity of all authentications paths in c using the PCP verifier. We refer to Section 5 for the full details.

2.2 Concurrent Zero-Knowledge

Let us first explain at a very high-level the challenges that we are faced with. In Barak's protocol [1], if one tries to obtain unbounded concurrency, the simulation overhead grows polynomially with every nested execution. The idea of Chung, Lin, and Pass [14] was to leverage **P**-certificates, i.e., succinct non-interactive proofs, to shortcut some of the nested computations. Particularly, **P**-certificates allowed them to reuse proofs that some computation was done correctly without the need to recompute it. This makes sense in the uniform setting where we assume that no false **P**-certificates can be found. However, using $\mathsf{nuCerts}$ in the non-uniform setting, it seems to be a problem because—intuitively—false $\mathsf{nuCerts}$ can be combined to get many more false proofs: a false proof that A implies B can be combined with a correct proof that B implies C to get a false proof that A implies C. The way we overcome this is by combining the proofs and the statements that we prove in a sequence such that if the adversary comes up with a false proof for one statement, it changes the entire sequence and forces the adversary to come up with many false proofs for new statements. In what follows, we discuss in detail the shortcomings with the protocols of Barak [1] and of Chung et al. [14], and then we explain how we avoid the aforementioned mix-and-match attack using $\mathsf{nuCerts}$.

Barak's protocol. We recall Barak's non-black-box constant-round zero-knowledge protocol [1] that achieves bounded concurrency. On common input 1^n and $x \in \{0,1\}^{\mathsf{poly}(n)}$, the Prover P and Verifier V, proceed in two phases. In phase 1, P sends a commitment $c \in \{0,1\}^n$ of 0^n to V, and V replies with a challenge $r \in \{0,1\}^{2n}$. In phase 2, P shows (using a witness indistinguishable argument of knowledge) that either x is true, or there exists a "short" string $\sigma \in \{0,1\}^n$ such that c is a commitment to a program M such that $M(\sigma) = r$. Soundness follows from the fact that even if a malicious prover P^* tries to commit to some program M (instead of committing to 0^n), with high probability, the string r sent by V will be different from $M(\sigma)$ for every string $\sigma \in \{0,1\}^n$. This relies on the fact that r is longer than σ. To prove ZK, consider the non-black-box simulator S that commits to the code of the malicious verifier V^*; note that by definition it thus holds that $M(c) = r$, and the simulator can use

$\sigma = c$ as a "fake" witness in the final proof. To formalize this approach, the witness indistinguishable argument in Stage 2 must be a witness indistinguishable universal argument (WIUA) [45,2] since the statement "c is a commitment to a program M of arbitrary polynomial size and $M(c) = r$ within some arbitrary polynomial time," is not in **NP**.

To show (bounded) concurrency, we need to simulate the view of a verifier that has $m = \mathsf{poly}(n)$ concurrent executions of the protocol. The above simulator no longer works in this setting: the problem is that the verifier's code is now a function of all the prover messages sent in different executions. So one solution is to increase the length of r in the above protocol to depend on the number of concurrent sessions, then we would be done by a similar argument. However, such an approach can handle only an *a priori* bounded number of sessions. A natural idea is to let the simulator commit not only to the code of V^*, but also to a program M that generates all other prover messages. Implementing this idea naively results with exponential blowup in the running time of the simulation since the verifier may nest concurrent sessions [49].

Uniform soundness via P-certificates. The main idea of Chung, Lin, and Pass [14] is to use **P**-certificates to overcome this blowup to achieve *unbounded* concurrency. At a very high level, their idea is that once the simulator has generated a **P**-certificate π to certify some partial computation performed by S in one session i, then the same certificate may be reused (without any additional cost) to certify the same computation also in other sessions $i' \neq i$, providing a "shortcut" for the simulator. It is crucial that the **P**-certificates are both fully non-interactive and succinct. Without the former, the certificates cannot be reused, and without the latter, we will not gain anything by reusing proofs.

Chung et al. [14] define a sequence of protocols $\Pi_1, \Pi_2, \ldots$, where protocol Π_k satisfies zero-knowledge for n^k concurrent sessions. The "trapdoor" for the simulation in Π_1, in contrast to Barak's protocol, now only requires that the a cheating prover can open the commitment to a machine M_1 and provide a **P**-certificate π_1 certifying that M_1 outputs the challenge r. However, we have not really gained anything over Barak's protocol yet since the challenge r can depend on the **P**-certificates in all previous sessions. Protocol Π_k uses k "levels" of **P**-certificates in a tree structure, where each higher level **P**-certificate certifies the correct generation of n lower level **P**-certificates. Then in protocol Π_k, the trapdoor requires that a cheating prover can open the commitment to a sequence of $M_1, \ldots, M_k$ such that: (1) for $i > 1$, the cheating prover can provide a **P**-certificate π_i certifying that M_i (given all higher level **P**-certificates) outputs the certified level $i - 1$ **P**-certificates, and (2) π_1 certifies that M_1 outputs r (again given all higher level **P**-certificates). The challenge r then only needs to depend on the non-certified **P**-certificates, which, because of the tree structure, is significantly smaller (and in particular apriori bounded) than the number of concurrent session.

Achieving non-uniform soundness via $\mathsf{nuCerts}$ for P. For security, the resulting concurrent zero-knowledge protocol shares the soundness guarantee of the underlying **P**-certificates. As we previously discussed, **P**-certificates can

only satisfy uniform soundness (under standard assumptions), so our approach to overcome this is to replace **P**-certificates with nuCerts for **P**, which guarantee non-uniform soundness at the cost of allowing the adversary to cheat on a bounded number of statements. To argue soundness, we need to use a cheating prover in the zero-knowledge protocol to extract many false nuCerts. We can rewind the cheating prover many times until we extract a large collection of accepting proofs for false statements. While seemingly simple, there are several technicalities with this argument.

First, the adversary may be able to use different false proofs per statement and mix-and-match statements to cheat in an exponential number of ways. In general, this indeed seems like an unavoidable problem. Chung et al. [14] suggested a way around it if one assumes a strong version of **P**-certificates, called *unique* **P**-certificates, which guarantee that every statement has *at most* one accepting proof. Using indistinguishablity obfuscation, Chung et al. [15] constructed a slightly weaker primitive, called delegatable unique **P**-certificates, and modified the protocol of [14] to work with this object (at the cost of making it private-coin).

We get around this by noticing that this is not a problem for us! In short, the thing that saves us is that we require the entire *sequence* of proofs to be certified (not just individually, thus the order matters). If the adversary tries to use a different false proof at any level, it yields an entirely new sequence that it must cheat on. In more detail, recall that the construction consists of a tree of nuCerts that certify the whole computation. At level 1 (the leaves), we have a sequence of n^k "certified" nuCerts λ_1. At level 2, we have a sequence of n^{k-1} nuCerts λ_2, and so on until level k when we don't have any certified nuCerts. As the name suggests, the role of each level i is to certify the total computation that happens below level i, and level 1 certifies the randomness r. So, π_1 is a nuCert certifying that there is a (deterministic) machine M_1 such that M_1 on input $\lambda_{\geq 1}$ outputs r, where $\lambda_{\geq 1}$ consists of all certificates at or above level 1. π_2 certifies that M_2 on input $\lambda_{\geq 2}$ outputs λ_1. π_3 certifies that M_3 on input $\lambda_{\geq 3}$ outputs λ_2, and so on until π_k, which certifies that M_k on a short input outputs λ_{k-1}.

The machines $M_1, \ldots, M_k$ are deterministic and are committed to by the adversary before seeing r. So, even though there could be a single certificate that explains multiple values of r, when we fix the machines ahead of time it can only output one of them. Namely, the adversary can come up with many inputs for M_1 that output any value of r just by having many "options" for the sequence λ_1. Since there are many choices for λ_1 and M_2 is deterministic, either the adversary has many accepting proofs π_2 for false statements that look like "M_2 on input $\lambda_{\geq 2}$ outputs λ_1" with a small set of $\lambda_{\geq 2}$, or the statements are in fact true and there are many different sequences $\lambda_{\geq 2}$ explaining them. In the former case, we are done as we can extract these proofs from the adversary. In the latter case, we must have many different sequences $\lambda_{\geq 2}$. We can continue with the same argument in level 3 that should output and explain all options for λ_2, and so on until the root of the tree at level k. At this point, we have that a short input is used to explain many possible random outputs, which is information

theoretically impossible unless the adversary knows many accepting proofs for false statements. We formalize this intuition by a compression argument, and we refer to Section 6 for the full details of the argument.

3 Preliminaries

For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. We use $\mathbb{N}$ to denote the set of positive integers and $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

We consider interactive Turing machines, denoted ITM, and interactive protocols. Given a pair of ITMs, A and B, we denote by $(A(x), B(y))(z)$ the random variable representing the (local) output of B on common input z and private input y, when interacting with A on private input x, when the random tape of each machine is uniformly and independently chosen. We also let $\mathsf{View}_B(A(x), B(y))(z)$ be the random variable representing B's view in such an interaction.

The term *negligible* is used for denoting functions that are asymptotically smaller than any inverse polynomial. We say that an event happens with *noticeable* probability if it happens with non-negligible probability, i.e. greater than $1/p(\cdot)$ probability for polynomial p. We say that an event happens with *overwhelming* probability if it occurs with all but negligible probability, i.e. at least $1 - \nu(\cdot)$ probability for negligible ν.

(Non)-uniformity. We use the acronym PPT for *probabilistic polynomial-time*. A uniform PPT machine can be thought of as a fixed Turing machine that has access to an input tape and performs some computation on the given input. If the computation is randomized, the machine has access to a random tape as well. A non-uniform machine can be thought of as a family of machines, one for each input length. Equivalently, one can think of a non-uniform machine as a single machine for all input lengths that has access to an advice string which might be different for every input length.

Witness relations for NP. We recall the definition of a witness relation for an **NP** language [27]. A *witness relation* for a language $L \in \mathbf{NP}$ is a binary relation $\mathbf{R}_L$ that is polynomially bounded, polynomial time recognizable, and characterizes L by $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathbf{R}_L\}$. We say that w is a witness for the membership $x \in L$ if $(x, w) \in \mathbf{R}_L$. We also let $\mathbf{R}_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e. $\mathbf{R}_L(x) = \{w : (x, w) \in \mathbf{R}_L\}$.

Commitments and collision resistant hashing. Commitment protocols allow a *sender* to commit itself to a value while keeping it secret from the *receiver*; this property is called *hiding*. At a later time, the commitment can only be opened to a single value as determined during the commitment protocol; this property is called *binding*. We consider non-interactive, computationally-hiding, statistically-binding commitment schemes. Such commitment schemes can be based on one-way functions [46,37] in the common random string (CRS) model,

but we ignore this for simplicity as the CRS can be generated honestly by the receiver.

We also consider families of collision resistant hash functions. A family of functions $F = \{F_n \colon \{0,1\}^{2n} \to \{0,1\}^n\}$ is collision resistant if for any non-uniform PPT adversary, the probability (over a random function in the family) that it can output a collision is negligible. It is known that a secure fixed-length hash function family can be used to obtain a secure variable-input-length hash function family, i.e., we can hash arbitrarily long inputs while guaranteeing collision resistance.

We say that a commitment scheme or a collision resistant hash function family is T-secure if every non-uniform $\mathsf{poly}(T)$-time attacker can break the corresponding security property with at most negligible in T probability.

3.1 Interactive Protocols

We define interactive proofs [30] and arguments systems (a.k.a. computationally sound proofs) [9]. In our definition of arguments, we distinguish between uniform soundness, where soundness only needs to hold against a uniform PPT adversary, and non-uniform soundness, where it holds against non-uniform polynomial-time algorithms. Typically, in the literature on zero-knowledge arguments, non-uniform soundness is more commonly used.

Definition 1 (Interactive proof system). *A pair of interactive machines (P, V) is called an* interactive proof system *for a language L if there is a negligible function $\nu(\cdot)$ such that the following two conditions hold:*

- Completeness*: For every $n \in \mathbb{N}$, $x \in L$ and every $w \in \mathbf{R}_L(x)$,*

$$\Pr[(P(w), V)(1^n, x) = 1] = 1.$$

- Soundness: *For every machine P^* and every $n \in \mathbb{N}$,*

$$\Pr[(x, z) \leftarrow P^*(1^n) : x \notin L \wedge (P^*(z), V)(1^n, x) = 1] \leq \nu(n).$$

If the soundness condition only holds against all non-uniform PPT (resp. uniform PPT) machines P^*, the pair (P, V) is called a non-uniformly sound (resp. uniformly sound) interactive argument system.

Witness indistinguishability. An interactive protocol is *witness indistinguishable* (WI) [21] if the verifier's view is "independent" of the witness used by the prover for proving the statement.

Definition 2 (Witness indistinguishability). *An interactive protocol (P, V) for $L \in \boldsymbol{NP}$ is* witness indistinguishable *for $\mathbf{R}_L$ if for every PPT adversarial verifier V^*, and for every two sequences $\{w^1_{n,x}\}_{n\in\mathbb{N},x\in L\cap\{0,1\}^{\mathsf{poly}(n)}}$ and $\{w^2_{n,x}\}_{n\in\mathbb{N},x\in L\cap\{0,1\}^{\mathsf{poly}(n)}}$ such that $w^1_{n,x}, w^2_{n,x} \in \mathbf{R}_L(x)$ for every $n \in \mathbb{N}$ and $x \in L \cap \{0,1\}^{\mathsf{poly}(n)}$, the following ensembles are computationally indistinguishable over $\mathbb{N}$:*

– $\{\mathsf{View}_{V^*}(P(w^1_{n,x}), V^*(z))(1^n, x)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, z\in\{0,1\}^*}$
– $\{\mathsf{View}_{V^*}(P(w^2_{n,x}), V^*(z))(1^n, x)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, z\in\{0,1\}^*}$

Universal arguments. Universal arguments, introduced by Barak and Goldreich [2], are used in order to provide "efficient" proofs to statements of the universal language $L_{\mathcal{U}}$ with witness relation $\mathbf{R}_{\mathcal{U}}$ defined in [2]. This notion is closely related to the notion of CS-proofs [45]. A triplet $q = (M, x, t)$ is in $L_{\mathcal{U}}$ if the non-deterministic machine M accepts input x within $t < T(|x|)$ steps, for a slightly super-polynomial function $T(n) = n^{\log\log n}$. We denote by $T_M(x, w)$ the running time of M on input q using the witness w. Notice that every language in **NP** is linear time reducible to $L_{\mathcal{U}}$. Thus, a proof system for $L_{\mathcal{U}}$ allows us to handle all **NP**-statements.

Definition 3 (Universal argument [2]). *A pair of interactive Turing machines (P, V) is called a* universal argument system *if it satisfies the following properties:*

– Efficient verification: *There exists a polynomial p_V such that for any $q = (M, x, t)$, the total time spent by the (probabilistic) verifier strategy V, on common input 1^n, q, is at most $p_V(n + |q|)$. In particular, all messages exchanged in the protocol have length smaller than $p_V(n + |q|)$.*
– Completeness by a relatively efficient prover: *For every $n \in \mathbb{N}$, $q = (M, x, t) \in L_{\mathcal{U}}$, and $w \in \mathbf{R}_{\mathcal{U}}(q)$,*

$$\Pr[(P(w), V)(1^n, q) = 1] = 1.$$

Furthermore, there exists a polynomial p_P such that the total time spent by $P(w)$, on common inputs 1^n and (M, x, t), is at most $p_P(n + |q| + t)$.
– Computational soundness: *For every non-uniform PPT algorithm $P^* = \{P^*_n\}_{n\in\mathbb{N}}$, there is a negligible function negl, such that, for every $n \in \mathbb{N}$ and every triplet $(M, x, t) \in \{0, 1\}^{\mathsf{poly}(n)} \setminus L_{\mathcal{U}}$,*

$$\Pr[(P^*_n, V)(1^n, q) = 1)] \leq \mathsf{negl}(n).$$

– Global proof of knowledge: *For every polynomial p_1 there exists a polynomial p_2 and a probabilistic oracle machine $E^{(\cdot)}$ such that the following holds: for every non-uniform PPT algorithm $P^* = \{P^*_n\}_{n\in\mathbb{N}}$, every sufficiently large $n \in \mathbb{N}$, and every $q = (M, x, t) \in \{0, 1\}^{\mathsf{poly}(n)}$, if $\Pr[(P^*_n, V)(1^n, q) = 1] \geq 1/p_1(n)$, then*

$$\Pr_r[\exists w \in \mathbf{R}_{\mathcal{U}}(q), {E_r}^{P^*_n}(1^n, q) = w] \geq 1/p_2(n),$$

where ${E_r}^{P^*_n}$ runs in time $\mathsf{poly}(n, t)$, uses randomness fixed to r, and has oracle access to P^*_n.

The notion of witness indistinguishability of universal argument for $\mathbf{R}_{\mathcal{U}}$ is defined similarly as that for interactive proofs/argument for **NP** relations; we refer the reader to [2] for a formal definition. [2] (based on [45,39]) presents a witness indistinguishable universal argument based on the existence of families of collision resistant hash functions.

3.2 List-Recoverable Codes

List-recoverable codes were introduced Guruswami and Sudan [34] to handle a setting where an adversary is allowed to submit a set $S \subseteq \mathbb{F}$ (where $\mathbb{F}$ is the alphabet of the code) of possible symbols and then construct any codeword using only those symbols. In this model, it is impossible to completely recover a codeword given the lists, but these codes guarantee that there is only a small list of codewords that are consistent with all the lists.

More precisely, a mapping $\mathsf{LRC}\colon \mathbb{F}^v \to \mathbb{F}^m$ from length v messages to length m codewords, is called (α, ℓ, L)-*list-recoverable* if there is a procedure that is given a set S of size ℓ, is able to output all messages $x \in \mathbb{F}^v$ such that $\mathsf{LRC}(x)_i \notin S$ for at most $1-\alpha$ fraction of the coordinates $i \in [n]$. The code guarantees that there are at most L such messages. For our purposes, we need a list-recoverable code with $\alpha = 1$, which we refer to as an (ℓ, L)-list-recoverable code, defined formally as follows.

Definition 4 (List-recoverable codes). *We say that a tuple $x \in (\{0,1\}^k)^m$ is* consistent *with sets $S_1, \ldots, S_m \subseteq \{0,1\}^k$ if $x_i \in S_i$ for all $i \in [m]$.*

A function $\mathsf{LRC}\colon \{0,1\}^v \to (\{0,1\}^k)^m$ is (ℓ, L)-list-recoverable, if for any sets $S_1, \ldots, S_m \subseteq \{0,1\}^k$ each of size at most ℓ, there are at most L strings $x \in \{0,1\}^v$ such that $\mathsf{LRC}(x)$ is consistent with $S_1, \ldots, S_m$. The strings in the image of LRC are referred to as codewords.

It is well-known (see e.g., [36]) that the notion of list-recoverable codes is equivalent to unbalanced expanders with a certain expansion property. The left set of vertices in the graph is $\{0,1\}^v$, the right set of vertices is $\{0,1\}^k$ and the left degree is m. This graph naturally induces a mapping $\mathsf{LRC}\colon \{0,1\}^v \to (\{0,1\}^k)^m$ which on input $x \in \{0,1\}^v$ (left vertex) outputs n neighbors (right vertices). The mapping LRC is (ℓ, L)-list-recoverable if and only if for every set $S \subseteq \{0,1\}^k$ of size larger than L of nodes on the right, the set of left neighbors of S is of size larger than ℓ.

The following instantiation of locally-recoverable codes based on the explicit construction of unbalanced expanders of [35] is taken (with minor modifications) from [36].

Theorem 3 ([35,36]). *For every $k < v \in \mathbb{N}$, there exists a $\mathsf{poly}(v)$-time computable function $\mathsf{LRC}_v\colon \{0,1\}^v \to \left(\{0,1\}^k\right)^{v^2k^3}$ that defines an (ℓ, L)-list-recoverable code for any $L \geq \ell^2$ such that $L \leq 2^{k/2}$. The list-recovery algorithm runs in time $\mathsf{poly}(v, \ell)$.*

3.3 Concurrent Zero-Knowledge

An interactive proof is said to be *zero-knowledge*, denoted as ZK, if it yields nothing beyond the validity of the statement being proved [30].

Definition 5 (Zero-knowledge). *An interactive protocol (P, V) for language L is* zero-knowledge *if for every PPT adversarial verifier V^*, there exists a PPT*

simulator S such that the following ensembles are computationally indistinguishable over $n \in \mathbb{N}$:

- $\{\mathsf{View}_{V^*}(P(w), V^*(z))(1^n, x)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, w\in\mathbf{R}_L(x), z\in\{0,1\}^{\mathsf{poly}(n)}}$
- $\{S(1^n, x, z)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, w\in\mathbf{R}_L(x), z\in\{0,1\}^{\mathsf{poly}(n)}}$

In this work, we consider the setting of concurrent composition. Given an interactive protocol (P, V) and a polynomial m, an m-session *concurrent adversarial verifier* V^* is a PPT machine that, on common input x and auxiliary input z, interacts with up to $m(|x|)$ independent copies of P concurrently. The different interactions are called sessions. There are no restrictions on how V^* schedules the messages among the different sessions, and V^* may choose to abort some sessions but not others. For convenience of notation, we overload the notation $\mathsf{View}_{V^*}(P, V^*(z))(1^n, x)$ to represent the view of the cheating verifier V^* in the above mentioned concurrent execution, where V^*'s auxiliary input is z, both parties are given common input $1^n, x \in L$, and the honest prover has a valid witness w of x.

Definition 6 (Concurrent zero-knowledge [19]). *An interactive protocol (P, V) for language L is* concurrent zero-knowledge *if for every concurrent adversarial verifier V^* (i.e. any m-session concurrent adversarial verifier for any polynomial m), there exists a PPT simulator S such that the following two ensembles are computationally indistinguishable over $n \in \mathbb{N}$:*

- $\{\mathsf{View}_{V^*}(P(w), V^*(z))(1^n, x)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, w\in\mathbf{R}_L(x), z\in\{0,1\}^{\mathsf{poly}(n)}}$
- $\{S(1^n, x, z)\}_{n\in\mathbb{N}, x\in L\cap\{0,1\}^{\mathsf{poly}(n)}, w\in\mathbf{R}_L(x), z\in\{0,1\}^{\mathsf{poly}(n)}}$

4 Non-Uniformly Sound Certificates

We give a definition of a certificate system that captures non-uniform attackers. Roughly speaking, a certificate system, (P, V), is a non-interactive (unidirectional) argument system (i.e., the prover send a single message to the verifier, who either accepts or rejects) such that (1) P can efficiently convince V the validity of some statement $x \in L$ using a "certificate" π of fixed polynomial length *independent* of the statement and (2) V can efficiently check the validity of π in fixed polynomial time in the statement size yet independent of the language's verification time. To capture non-uniform attackers, we additionally require (3) that no non-uniform cheating prover P^* should be able to falsely convince V of the validity of substantially more statements than the size of P^*'s non-uniform advice. In what follows, we first formalize these requirements for $\mathbf{P}$ and later discuss (in Remark 1) how to generalize them to $\mathbf{NP}$.

Following Micali's CS-proofs [45], we first define the efficiency properties required for any certificate system for $\mathbf{P}$. We consider a canonical language L_c for $\mathsf{TIME}(n^c)$: for every constant $c \in \mathbb{N}$, let L_c be the language that consists of triples (M, x, y) such that machine M on input x outputs y when executed for $|x|^c$ steps. That is,

$$L_c = \{(M, x, y) \colon M(x) = y \text{ within } |x|^c \text{ steps}\}.$$

Let $T_M(x)$ denote the running time of M on input x.

Definition 7 (Certificates for P). *Let (P, V) be a pair of probabilistic interactive Turing machines in a non-interactive protocol. We say that (P, V) is a* certificate system for $\mathbf{P}$ *if it satisfies the following two efficiency conditions:*

- Completeness by a relatively-efficient prover: *There exist polynomials g_P, ℓ such that for every $c, n \in \mathbb{N}$ and every $q = (M, x, y) \in L_c$, it holds that*

$$\Pr[\pi \leftarrow P(c, 1^n, q) : V(c, 1^n, q, \pi) = 1] = 1.$$

  Furthermore, P on input $(c, 1^n, q)$ outputs a certificate π of length $\ell(n)$ in time bounded by $g_P(n + |q| + T_M(x))$.

- Efficient verification: *There exists a polynomial g_V such that for every $c, n \in \mathbb{N}$, $q = (M, x, y) \in L_c$ and $\pi \in \{0,1\}^*$, the running time of $V_{\mathsf{nu}}(c, 1^n, q, \pi)$ is bounded by $g_V(n + |q|)$.*

Best-possible soundness. At a high level, we require non-uniformly sound certificates to achieve the *best-possible soundness* against non-uniform attackers. This means that our notion of soundness allows the adversary to come up with *some* accepting certificates for false statements, but not too many more than the size of its advice. We formalize this intuition with the notion of (K, T)-soundness, which intuitively says that a non-uniform adversary running in time T cannot output false proofs for more than K statements. We define this as follows.

Definition 8 ((K, T)-soundness). *Let $K, T \colon \mathbb{N} \to \mathbb{N}$ be functions. We say that a certificate system for $\boldsymbol{P}$, (P, V), is* (K, T)-sound *if there exists a function $B(n) \in \omega(1)$ such that the following holds:*
For every probabilistic algorithm P^* and any sequence of polynomial-size advice $\{z_n\}_{n \in \mathbb{N}}$ where $P^*(1^n, z_n)$ runs in time at most $T(n)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$, letting $K = K(n)$,

$$\Pr\left[\{(c_i, q_i, \pi_i)\}_{i \in [K]} \leftarrow P^*(1^n, z_n) : \begin{array}{c} \forall i \neq j \colon q_i \neq q_j, \\ \forall i \in [K] \colon c_i \leq B(n) \wedge \\ q_i \notin L_{c_i} \wedge \\ V(c_i, 1^n, q_i, \pi_i) = 1 \end{array}\right] \leq \mathsf{negl}(n).$$

We are now ready to define non-uniformly sound certificates ($\mathsf{nuCerts}$) for $\mathbf{P}$. Intuitively, we require that no non-uniform cheating prover with polynomial-size advice can output a super-polynomial number of accepting proofs for false statements where we allow the cheating prover to run in super-polynomial time per false proof it outputs.

Definition 9 ($\mathsf{nuCerts}$ for P). *Let (P, V) be a pair of probabilistic interactive Turing machines in a non-interactive protocol. We say that (P, V) is a* non-uniformly sound certificate system (nuCert) for $\mathbf{P}$ *if it is a certificate system for $\boldsymbol{P}$ and is (K, T)-sound for some $K(n) \in n^{\omega(1)}$ and $T(n) \in K(n) \cdot n^{\omega(1)}$.*

Some remarks are in order.

Remark 1. *The above definition generalizes to* ***NP*** *by considering non-deterministic languages L_c corresponding to* $\mathsf{NTIME}(n^c)$ *where we provide the prover with a witness w and let $T_M(x)$ be the time to verify that $(x, w) \in \mathbf{R}_L$.*

Remark 2. *The notion of (K, T)-sound certificate systems for* ***P*** *is a generalization of strong* ***P****-certificates (of Chung et al. [14]) as they coincide when the number of false proofs is just one, i.e., $K(n) = 1$, and the running time of the adversary is a (slightly) super-polynomial function, i.e., $T(n) \in n^{\omega(1)}$.*

Remark 3. *The definition of (K, T)-soundness as defined can only be achieved for super-polynomial $K(n) \in n^{\omega(1)}$. This notion is sufficient for our concurrent zero-knowledge protocol given in Section 6. However, one can consider a more fine-grained notion where we allow the number of false proofs to depend on the size of the advice and accepting statements. We actually achieve this (stronger) notion in our* nuCerts *construction in Section 5.*

Remark 4. *The size of accepting statements $q = (M, x, y)$ for L_c must be bounded by $O(|x|^c)$ without loss of generality since V would not accept q for the language L_c if $|y| > |x|^c$.*

Remark 5. *We could consider an alternative definition of (K, T)-soundness where we remove the restriction that statements are unique and just require that the proofs are. These notions are equivalent up to a factor that depends exponentially on the certificate size for K.*

In more detail, one trivial way to bound the number of false proofs in general is to bound the number of false proofs for specific statements. In the worst case, if you can find a false proof for an accepting statement, then all proofs for that statement may be accepted. So, if a (K, T)-sound certificate system has proofs π with length bounded by $\ell_\pi(n)$, this implies it is $(K \cdot 2^{\ell_\pi(n)}, T)$-sound when the statements need not be unique.

Relation to Kolmogorov complexity. Our defintion has a natural interpretation in the language of computational Kolmogorov complexity. Recall that the Kolmogorov function $C(x)$ is the length of the smallest program generating x. Namely, $C(x) = \min_p\{|p| \colon U(p) = x\}$, where U is a universal machine. There are many resource-bounded variants to Kolmogorov comlexity [23]. Our definition is parametrized by an efficiently recognizable set X and a time bound t. We define $C^t(X)$ as the smallest machine that runs in time t and outputs an element in X. Namely,

$$C^t(X) = \min_p\{|p| \colon U(p) \text{ outputs } x \in X \text{ in } t(|x|) \text{ steps}\}.$$

Claim 1. *Let (P, V) be a certificate system, $K \in n^{\omega(1)}$, $T \in K(n) \cdot n^{\omega(1)}$, and let X be a set of sets such that every set is a collection of accepting proofs for K false statements. (P, V) is (K, T)-sound if and only if $C^T(X) \in n^{\omega(1)}$.*

Proof. Assume for contradiction that $C^T(X) \in \mathsf{poly}(n)$. This means there is a polynomial size encoding that in time T can be used to generate K accepting proofs for false statements, contradicting (K,T)-soundness of (P,V). In the other direction, assume that (P,V) satisfies (K,T)-soundness. Then, there is a cheating prover with polynomial-size advice running in time T that is able to find K accepting proofs for false statements. This implies a machine of polynomial size that encodes an element in X, contradicting $C^T(X) \in n^{\omega(1)}$.

5 The Construction

In this section, we give a construction of a nuCert system for **P** as defined in Section 4. Our construction is in the auxiliary input random oracle model, introduced by Unruh [55]. In this model all parties have access to a public random function. Additionally, the adversary has an unbounded preprocessing stage (the offline phase) where he can compute arbitrarily an advice string for the online phase where he attacks the system. The output of the offline phase can be thought of as the process that generates the non-uniform advice. In the online phase, the adversary can use the advice string and a bounded number of queries (though his running time is unbounded).

Theorem 4. *In the auxiliary-input random oracle model, there exists a certificate system for* $\boldsymbol{P}$ *that satisfies* (K,T)*-soundness for any* $K(n) \geq (3s)^{25\alpha}$ *and* $T(n) \leq 2^{n/6}$ *against non-uniform adversaries with advice* $\{z_n\}_n$ *of size* $s = s(n)$ *that output accepting statements of size at most* n^{α}*.*

The protocol we present is actually a nuCert for **NP** (with essentially the same proof). In Section 6, we will use the result only for **P** so we focus on this setting here.

We note that if the adversary outputs statements of super-polynomial size, we may assume that they will not be accepted by a verifier for the universal language for **P**. Still, the adversary may output accepting statements of arbitrary polynomial size, so without assuming anything about the adversary, the theorem holds for any slightly super-constant $\alpha \in \omega(1)$. In this case, we guarantee that no adversary can output a slightly super-polynomial number of accepting proofs for unique false statements, e.g., $K(n) = n^{\log\log n}$. Additionally, we can set $T(n) = n^{\log n}$ so that $T(n) \in K(n) \cdot n^{\omega(1)}$. Thus, we get the following corollary to Theorem 4 by Definition 9.

Theorem 5. *[Restatement of Theorem 1] In the auxiliary-input random oracle model, there exists a* nuCert *system for* $\boldsymbol{P}$*.*

We devote the rest of this section to the proof of Theorem 4 which is based on a compression technique (à la Gennaro and Trevisan [25]). In Section 5.1 we describe our nuCert construction and then prove security in the auxiliary-input random oracle model in Section 5.2.

5.1 Construction

Our nuCert system, denoted $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$, builds off of Micali's CS-proofs but makes significant modifications. Recall that in Micali's protocol, the prover uses a random oracle to compute a Merkle hash a of a PCP proof for the statement q. The prover then uses a random oracle to compute b, defining a set of PCP challenge queries. Finally, it computes the authentication paths c in the Merkle tree for all openings in the PCP proof specified by b. The final proof $\pi = (a, b, c)$ is sent to the verifier to be checked.

Merkle trees and PCPs. In the construction, we will use Merkle trees and probabilistic checkable proofs (PCPs).

A Merkle tree [44] is a method to succinctly commit on a string while allowing to open a specific location without revealing the whole input string. The latter property is called *local opening*. The security of the commitment is based on the existence of a collision resistant hash function family. More precisely, a Merkle tree procedure $\mathsf{MT}^{f(\cdot)}$ has oracle access to a compressing function $f\colon \{0,1\}^{\lambda} \to \{0,1\}^{n}$, which it uses to hash a long string into a short one. This is done by "breaking" the input into blocks of the right size and hasing each consecutive pair using the hash function. We continue this recursively in a tree-like fashion until we are left with a single string of size n.

A PCP system is a proof system that allows for local verification of a language. It consists of two algorithms $(\mathsf{PCP.Prove}, \mathsf{PCP.Ver})$. $\mathsf{PCP.Prove}(x)$ is a deterministic algorithm that takes as input a statement x from some language L and computes a proof Π to the fact that $x \in L$. $\mathsf{PCP.Ver}^{\Pi}(x, r)$ is a randomized algorithm using r as its randomness source that takes as input a statement x and has query access to a (possibly partial) PCP proof Π and outputs a bit b.

The construction. We split $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$ into two different phases: a commitment and a proof phase. The commitment phase forces P_{nu} to succinctly commit to the statement it is proving. The proof phase is exaclty as Micali's CS-proofs with the modification that the random oracles used are indexed by the statement's commitment.

In more detail, the commitment consists of two parts. The first part, which we denote by $\tilde{q}$, is a Merkle hash of a list-recoverable code encoding of q. The list-recoverable code provides a weak binding property for any particular commitment; no non-uniform adversary can find too many statements consistent with $\tilde{q}$. In the second part of the commitment, which we denote by $\hat{q}$, the prover Merkle hashes q using a random oracle indexed by $\tilde{q}$. $\hat{q}$ guarantees that a cheating prover that provides an accepting proof of a false statement q either uses specific knowledge that depends on $\tilde{q}$ or can be used to extract q.

The proof phase computes $\pi = (a, b, c)$ as in Micali's CS-proofs except that we index both the Merkle tree and FS-heuristic random oracles with the commitment $(\tilde{q}, \hat{q})$. This guarantees that each false proof with a unique commitment requires the adversary to use fresh information about the random oracle. The full description of the non-interactive protocol, $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$, is given in Figure 1.

A nuCert System $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$

Common Input: Security parameter 1^n, time bound c, and a statement $q = (M, x, y)$ for the language L_c.

Common Oracles: $\mathcal{O}_1 \colon \{0,1\}^{n^2} \to \{0,1\}^n$, $\mathcal{O}_2 \colon \{0,1\}^{n+n^2} \to \{0,1\}^n$, $\mathcal{O}_3 \colon \{0,1\}^{2n+n^2} \to \{0,1\}^n$, $\mathcal{O}_4 \colon \{0,1\}^{3n} \to \{0,1\}^{\ell_b}$.

Subroutines: PCP system $\mathsf{PCP} = (\mathsf{Prove}, \mathsf{Ver})$ with soundness error 2^{-n} using $\ell_b \in O(n \log n)$ bits of randomness, a Merkle tree algorithm $\mathsf{MT}^{(\cdot)}$, and collection of codes $\mathsf{LRC}_v \colon \{0,1\}^v \to (\{0,1\}^{n^2})^{v^2 n^6}$ that satisfies (ℓ, L)-list-recoverability for any $L \leq 2^{n^2/2}$ such that $L \geq \ell^2$.

Prover $P_{\mathsf{nu}}(1^n, q)$**:**

Commitment phase:

1. Compute $\tilde{q} \leftarrow \mathsf{MT}^{\mathcal{O}_1(\cdot)}(\mathsf{LRC}_{|q|}(q))$.
2. Compute $\hat{q} \leftarrow \mathsf{MT}^{\mathcal{O}_2(\tilde{q},\cdot)}(q)$.

Proof phase:

3. Compute $\Pi \leftarrow \mathsf{PCP.Prove}(q)$. Compute $a \leftarrow \mathsf{MT}^{\mathcal{O}_3(\tilde{q},\hat{q},\cdot)}(\Pi)$.
4. Compute $b \leftarrow \mathcal{O}_4(\tilde{q}, \hat{q}, a)$.
5. Let I be the set of indices queried by $\mathsf{PCP.Ver}^{\Pi}(q; b)$. Let c be the authentication paths for indices I in the Merkle tree of Π.
6. Send $\pi = (a, b, c)$ to V_{nu}.

Verifier $V_{\mathsf{nu}}(1^n, q, \pi)$**:**

Commitment verification:

1. Compute $\tilde{q} \leftarrow \mathsf{MT}^{\mathcal{O}_1(\cdot)}(\mathsf{LRC}_{|q|}(q))$.
2. Compute $\hat{q} \leftarrow \mathsf{MT}^{\mathcal{O}_2(\tilde{q},\cdot)}(q)$.

Proof verification:

3. Parse $\pi = (a, b, c)$. Verify $b = \mathcal{O}_3(\tilde{q}, \hat{q}, a)$.
4. Construct a partial PCP proof Π with openings from c.
5. Let I be the set of indices queried by $\mathsf{PCP.Ver}^{\Pi}(q; b)$. Verify the authentication paths c given the indices I and the Merkle hash root a.
6. Accept if and only if $\mathsf{PCP.Ver}^{\Pi}(q; b)$ accepts.

Fig. 1. A nuCert for $\mathbf{P}$ defined in the auxiliary-input random oracle model.

Efficiency and completeness. We first argue that $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$ is a valid certificate system for $\mathbf{P}$ by showing the *completeness by a relatively efficient prover* and *efficient verification* properties.

Completeness follows from the completeness of the underlying PCP proof system. For prover efficiency, note that computing $\tilde{q}$ and $\hat{q}$ in the commitment phase only takes time polynomial in the statement size. This uses the efficiency of the list-recoverable code. Recall that a proof $\pi = (a, b, c)$ consists of a n bit Merkle tree root, the randomness for the PCP proof, and the authentication paths for locations specified by the PCP proof. In order to compute a, the prover computes the PCP proof for q, which takes time polynomial in the machine's running time. Since the protocol uses a PCP proof system with soundness error 2^{-n}, the prover has to open and authenticate $O(n)$ bits using $O(\log n)$ bits of randomness per location, which takes polynomial time only in the security

parameter n. Each authentication path in the Merkle tree has size $O(n \log n)$. This implies a polynomial length bound of $\ell_\pi(n) \in O(n^2 \log n)$.

The verifier also needs to compute $\tilde{q}$ and $\hat{q}$, but does not need to compute the PCP proof for q. Thus, the verifier's running time is independent of the machine's running time, and depends only polynomially on the statement size $|q|$ and the proof length $\ell_\pi(n)$.

5.2 Proof of Theorem 4

The proof of (K, T)-soundness proceeds in two steps. We first show that a non-uniform adversary can only find a bounded number of statements per $\tilde{q}$ value in the commitment. Then we show that a non-uniform adversary cannot come up with a false proof for too many $\tilde{q}$ values.

Bounding statements per $\tilde{q}$. Recall that $\tilde{q}$ is a Merkle hash of a list-recoverable code encoding of some statement q. We show that a non-uniform cheating prover cannot find too many statements that yield the same $\tilde{q}$ value.

Lemma 1. *Let $\mathcal{O}$ be a random function from n^2 to n bits. For any functions $K, s\colon \mathbb{N} \to \mathbb{N}$ and sequence of advice $\{z_n\}_{n\in\mathbb{N}}$ of size $s = s(n) \geq 4$, let $\mathcal{A}^{\mathcal{O}}$ be a non-uniform algorithm that on input $(1^n, z_n)$ makes at most $2^{n/2}$ queries and outputs $K = K(n)$ unique statements $q_1, \ldots, q_K$. Define $N = (3s)^{24\cdot\alpha}$ where α satisfies $|q_i| \leq n^\alpha$ for all $i \in [K]$, and define $\tilde{q}_i = \mathsf{MT}^{\mathcal{O}(\cdot)}(\mathsf{LRC}_{|q_i|}(q_i))$. Then, for every $n \in \mathbb{N}$, it holds that*

$$\Pr_{\mathcal{O}}\left[(q_1, \ldots, q_K) \leftarrow \mathcal{A}^{\mathcal{O}}(1^n, z_n) : \begin{array}{c} \forall i \neq j \in [K]\colon q_i \neq q_j \\ |\{\tilde{q}_i : i \in [K]\}| \leq K/N \end{array}\right] \leq 2^{-n}.$$

The proof of this lemma appears in the full version.

Bounding false proofs per $\tilde{q}$. We show that if a cheating prover comes up with a false proof for a statement with a new $\tilde{q}$ value, it must have used some knowledge to either find (1) a new collision, (2) a new pre-image of a previously queried point, or (3) a new challenge message b that comes from a small set. We use each scenario to compress the random oracle's description on some input, which bounds the number of oracles a cheating prover can succeed on. We note that we can use a single oracle $\mathcal{O}$ with large enough domain and range to represent all four oracles defined in Figure 1. We can simply modify each query to first specify an index for which oracle it wants to query and then restrict the input and output to the correctly defined length. We formalize this compression-style argument in the following lemma.

Lemma 2. *Let $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$ be the* nuCert *system from Figure 1 with random oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_4)$. For any functions $K, s\colon \mathbb{N} \to \mathbb{N}$ and sequence of advice $\{z_n\}_{n\in\mathbb{N}}$ of size $s = s(n)$, let $\mathcal{A}^{\mathcal{O}}$ be a non-uniform algorithm that on input $(1^n, z_n)$ makes T oracle queries and outputs $(C_1, q_1, \pi_1), \ldots, (C_K, q_K, \pi_K)$ where*

$K = K(n)$. *Then, for every* $n \in \mathbb{N}$, *it holds that*

$$\Pr_{\mathcal{O}}\left[\{(C_i, q_i, \pi_i)\}_{i\in[K]} \leftarrow \mathcal{A}^{\mathcal{O}}(1^n, z_n) : \begin{array}{c} \forall i \neq j \in [K]\colon \tilde{q}_i \neq \tilde{q}_j \\ \wedge\, q_i \neq q_j, \\ \forall i \in [K]\colon q_i \notin L_{C_i} \\ \wedge\, V_{\mathsf{nu}}(C_i, 1^n, q_i, \pi_i) = 1 \end{array}\right] \leq 2^{-K(n-3\log T)+s}.$$

Proof. For each proof that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ outputs, parse $\pi_i = (a_i, b_i, c_i)$. We assume without loss of generality that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ makes every query that V_{nu} checks exactly once. Otherwise, we can modify $\mathcal{A}^{\mathcal{O}}$ to make all such queries at the end and never make the same query twice, which uses at most $K \cdot \mathsf{poly}(n)$ extra queries by the efficiency of V_{nu}. Let $Q_1, \ldots, Q_T$ be the unique random oracle queries that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ makes.

Before explicitly defining our representation of $\mathcal{O}$, we introduce some notation. For every statement q and PCP proof Π, we define $B(q, \Pi)$ to be the set of all b values for which $\mathsf{PCP.Ver}^{\Pi}(q, b)$ accepts, i.e.,

$$B(q, \Pi) = \{b \mid \mathsf{PCP.Ver}^{\Pi}(q; b) = 1\}.$$

By the 2^{-n} soundness error of the underlying PCP proof system, $|B(q, \Pi)| \leq 2^{\ell_b - n}$ for all statements q not in the specified language where recall ℓ_b is the length of b. When we refer to the mth element of the set $B(q, \Pi)$, we mean the mth element in the lexicographic enumeration of the set, which is uniquely defined.

We now define a procedure to generate a representation of a particular random oracle $\mathcal{O}$ using $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$. We assume for simplicity that z_n starts with the description of $\mathcal{A}$.

1. Initialize lists L_{col}, L_{pre}, L_{pcp}, L_{query}, and L_{other} to be empty.
2. Run $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ to get false proofs $\pi_i = (a_i, b_i, c_i)$ for statements q_i with unique $\tilde{q}_i$ values for $i \in [K]$.
3. For each query Q_j in order, do exactly one the following.
   (a) If $\mathcal{O}(Q_j) = \mathcal{O}(Q_m)$ for $m < j$, add (j, m) to L_{col}.
   (b) If $\mathcal{O}(Q_j) = Q_m[\mathsf{pos}\cdot n+1 : (\mathsf{pos}+1)\cdot n]$ (where $Q_m[i_1 : i_2]$ is the substring from index i_1 to i_2 inclusive) for $m < j$ and $\mathsf{pos} \in \{0, \ldots, n+1\}$, add (j, m, pos) to L_{pre}.
   (c) If the following conditions hold, add (j, m) to L_{pcp}.
      i. $Q_j = (\tilde{q}, \hat{q}, a_i)$ and $\mathcal{O}_4(\tilde{q}, \hat{q}, a_i) = b_i$ for some $i \in [K]$.
      ii. It is possible to uniquely extract a statement q that Merkle hashes to $\hat{q}$ using only previous $\mathcal{O}_2(\tilde{q}, \cdot)$ queries.
      iii. It is possible to uniquely extract a partial PCP proof Π that Merkle hashes to a_i using only previous $\mathcal{O}_3(\tilde{q}, \hat{q}, \cdot)$ queries.
      iv. b_i is the mth element of $B(q_i, \Pi)$.
   (d) Otherwise, add $\mathcal{O}(Q_j)$ to L_{query}.
4. For all other inputs x, add $\mathcal{O}(x)$ to L_{other} in lexicographic order.
5. Output z_n, L_{col}, L_{pre}, L_{pcp}, L_{query}, L_{other} as the representation.

Using only the representation, we define a procedure to compute all $\mathcal{O}$ queries.

1. Run $\mathcal{A}^{(\cdot)}(1^n, z_n)$ and simulate its queries to $\mathcal{O}$.
2. On the jth query Q_j, do the following.
   (a) If (j, m) is in L_{col} for $m < j$, output $\mathcal{O}(Q_m)$.
   (b) If (j, m, pos) is in L_{pre} for $m < j$ and $\mathsf{pos} \in \{0, \ldots, n+1\}$, output $Q_m[\mathsf{pos} \cdot n + 1 : (\mathsf{pos}+1) \cdot n]$.
   (c) If (j, m) is in L_{pcp}, do the following.
      i. Parse $Q_j = (\tilde{q}, \hat{q}, a)$.
      ii. Extract a statement q that Merkle hashes to $\hat{q}$ using only previous queries to $\mathcal{O}_2(\tilde{q}, \cdot)$.
      iii. Extract a partial PCP proof Π that Merkle hashes to a using only previous queries to $\mathcal{O}_3(\tilde{q}, \hat{q}, \cdot)$.
      iv. Output the mth element of $B(q, \Pi)$.
   (d) Otherwise, output the next value in the list L_{query}.
3. Compute all other outputs for $\mathcal{O}$ using L_{other}.

In the following claim, we prove that for any machine $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$, the above procedure defines a valid and unique representation for $\mathcal{O}$.

Claim 2. *For every machine $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$, the representation of $\mathcal{O}$ defined above is correct and unique.*

The proof of this claim appears in the full version.

We next show that the representation of $\mathcal{O}$ is actually compressing when $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ succeeds. Note that all random oracle outputs are added to exactly one of the lists L_{col}, L_{pre}, L_{pcp}, L_{query}, or L_{other}. Any output that is added to L_{col}, L_{pre}, or L_{pcp} requires fewer bits to represent than the full output. At a high level, we argue that at least K outputs will be added to these lists.

Claim 3. *Suppose that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ outputs a false proof $\pi = (a, b, c)$ for a statement q mapping to $\tilde{q}$. Then, there is some query unique to $\tilde{q}$ that was added to either L_{col}, L_{pre}, or L_{pcp} in the representation defined above.*

Proof. Suppose there is no query of the form $(\tilde{q}, \cdot)$ or $(\tilde{q}, \hat{q}, \cdot)$ added to L_{col} or L_{pre}. It suffices to prove there is some query Q_j of the form $(\tilde{q}, \hat{q}, a)$ that is added to L_{pcp}. Recall that we assume $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ queries everything checked by V_{nu}. This implies that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ queries $Q_j = (\tilde{q}, \hat{q}, a)$ at some point. We show that each of the conditions required for Q_j to be added to L_{pcp} hold. Namely, it is possible to uniquely extract q and Π such that $b = \mathcal{O}(\tilde{q}, \hat{q}, a)$ is contained in $B(q, \Pi)$.

First suppose it is not possible to uniquely extract q using previous queries to $\mathcal{O}_2(\tilde{q}, \cdot)$. We have assumed the entire Merkle tree computing $\mathsf{MT}^{\mathcal{O}_2(\tilde{q}, \cdot)}(q)$ is queried at some point. If we are not able to extract the correct statement q, then some query in the Merkle tree must be made after query Q_j. This would require that some pre-image will be queried using $\mathcal{O}_2(\tilde{q}, \cdot)$, which contradicts the assumption that no query containing $\tilde{q}$ is added to L_{pre}. If it is possible to extract the statement q but it is not unique, then there must be some collision queried in the Merkle tree using $\mathcal{O}_2(\tilde{q})$, but this contradicts that no query of the

form $(\tilde{q}, \cdot)$ was added to L_{col}. Thus, we must be able to uniquely extract q when query Q_j is made.

Similarly, we must be able to use previous queries to $\mathcal{O}_3(\tilde{q}, \hat{q}, \cdot)$ to uniquely reconstruct a partial PCP proof Π. If it is not unique, there must be a collision in $\mathcal{O}_3(\tilde{q}, \hat{q}, \cdot)$, which is a contradiction. If all authenticating paths are not queried before Q_j, there will be some pre-image queried with $\mathcal{O}_3(\tilde{q}, \hat{q}, \cdot)$, which again is a contradiction.

Lastly, because (a, b, c) is an accepting proof for the statement q, b must be in the set $B(q, \Pi)$, which has bounded size since q is not in the specified language. We conclude that Q_j must be added to L_{pcp}.

By the above claim, we add at least one query to L_{col}, L_{pre}, or L_{pcp} for each unique $\tilde{q}$ value. Each query added to L_{col} uses at most $2 \log T$ bits to represent, L_{pre} uses at most $2 \log T + \log(n+2)$ bits, and L_{pcp} uses at most $\log T + \ell_b - n$ bits to represent a value of size ℓ_b. Since $A^{\mathcal{O}}$ queries everything that V_{nu} queries, T is at least $n+2$, so $\log(n+2) \leq \log T$. This implies that we compress by at least $n - 3 \log T$ bits per query added to L_{col}, L_{pre}, or L_{pcp}. The total representation in bits in this case is at most

$$n \cdot (2^{2n} + 2^{3n} + 2^{4n}) + \ell_b \cdot 2^{3n} + s - K(n - 3 \log T).$$

Because the representation is unique, this bounds the total number of oracles that $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ succeeds by $2^{n \cdot (2^{2n} + 2^{3n} + 2^{4n}) + \ell_b \cdot 2^{3n} + s - K(n - 3 \log T)}$ but there are $2^{n \cdot (2^{2n} + 2^{3n} + 2^{4n}) + \ell_b \cdot 2^{3n}}$ oracles in total. So the probability $\mathcal{A}^{\mathcal{O}}(1^n, z_n)$ succeeds on a randomly chosen oracle is at most $2^{-K(n - 3 \log T) + s}$.

We finish the proof of Theorem 4 next.

Proof (of Theorem 4). Let $P^{*\mathcal{O}}(1^n, z_n)$ be a non-uniform cheating prover that makes T queries to the random oracle and outputs $(C_1, q_1, \pi_1), \ldots, (C_K, q_K, \pi_K)$ for unique statements. We will show that

$$\Pr_{\mathcal{O}}\left[\{(C_i, q_i, \pi_i)\}_{i \in [K]} \leftarrow P^{*\mathcal{O}}(1^n, z_n) : \begin{array}{c} \forall i \neq j \in [K] \colon q_i \neq q_j, \\ \forall i \in [K] \colon q_i \notin L_{C_i} \\ \wedge\, V_{\mathsf{nu}}(C_i, 1^n, q_i, \pi_i) = 1 \end{array}\right] \leq 2^{-n+1}.$$

Let SUCC be the event that $P^{*\mathcal{O}}(1^n, z_n)$ succeeds, i.e., the condition above holds.

Consider an execution of $P^{*\mathcal{O}}(1^n, z_n)$ for a random function $\mathcal{O}$ that outputs $(C_1, q_1, \pi_1), \ldots, (C_K, q_K, \pi_K)$. Define α to be the smallest constant such that such that $|q_i| \leq n^\alpha$ for all $i \in [K]$, and define $N = (3s)^{24 \cdot \alpha}$. Let BAD be the event that $|\{\tilde{q}_i = \mathsf{MT}^{\mathcal{O}}(\mathsf{LRC}_{|q_i|}(q_i)) : i \in [K]\}| \leq K/N$. Since $\Pr[\mathsf{SUCC}] \leq \Pr[\mathsf{BAD}] + \Pr[\mathsf{SUCC} \mid \neg\mathsf{BAD}]$, it suffices to bound each term separately.

By Lemma 1, $\Pr[\mathsf{BAD}] \leq 2^{-n}$. Given that BAD does not occur, we can find a set of statements $X \subseteq \{q_1, \ldots, q_K\}$ with false proofs and with a unique $\tilde{q}$ value where $|X| = K/N$. By Lemma 2, this can succeed with probability at most $2^{-(K/N) \cdot (n - 3 \log T) + s}$ over the choice of $\mathcal{O}$. When $K/N = 3s$ and $T \leq 2^{n/6}$, this is at most 2^{-n}. In this case, $K = (3s)^{24\alpha + 1} \leq (3s)^{25 \cdot \alpha}$, and $\Pr[\mathsf{SUCC}] \leq 2^{-n+1}$.

Remark 6. *As discussed in Remark 5, Theorem 4 immediately implies a corollary for a definition of (K, T)-soundness where the statements do not need to be unique. Specifically, we argued in Section 5.1 that $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$ has a proof length bound of $\ell_\pi = O(n^2 \log n)$. It follows that $(P_{\mathsf{nu}}, V_{\mathsf{nu}})$ is (K', T)-sound for $K' = (3s)^{25\alpha} \cdot n^{n^2}$ in the case where only the proofs need to be unique rather than the statements.*

6 Concurrent Zero-Knowledge Protocol

In this section, we present our concurrent zero-knowledge protocol and prove that it satisfies *non-uniform* soundness when instantiated with our notion of $\mathsf{nuCerts}$ for $\mathbf{P}$.

Theorem 6 (Restatement of Theorem 2). *Suppose there exist super-polynomially-secure families of collision resistant hash functions and $\mathsf{nuCerts}$ for $\boldsymbol{P}$. Then, there exists a public-coin constant-round concurrent zero-knowledge protocol for $\boldsymbol{NP}$ with non-uniform soundness and an explicit simulator.*

We recall from Theorem 5 that we construct a nuCert for $\mathbf{P}$ in the auxiliary-input random oracle model. By a similar compression argument as in Section 5, it is easy to show that a compressing random oracle is a secure collision resistant hash function even in the auxiliary-input random oracle model. Thus, we get the following corollary to Theorems 5 and 6.

Corollary 2 (Restatement of Corollary 1). *In the auxiliary-input random oracle model, there exists a public-coin constant-round concurrent zero-knowledge protocol for $\boldsymbol{NP}$.*

Our protocol is very similar to that of Chung et al. [14] which, in turn, is a variant of Barak's [1] constant-round (bounded concurrency) zero-knowledge protocol. At a very high level, Barak's simulator uses the verifier's code as a trapdoor to convince the verifier of the validity of the statement at hand. The simulator of Chung et al. instead uses to a succinct proof to certify that it knows a program that does the required task. To be able to provide such proofs for concurrent verifiers that may start nested sessions, rather than redoing computation multiple times, they design a way to provide succinct proofs to "shortcut" some of the computation. Specifically, they use k levels of uniformly sound certificates for $\mathbf{P}$ ($\mathbf{P}$-certificates). The first layer of $\mathbf{P}$-certificates is used to certify the verifier's messages in its interaction with the prover, and all above layers certify the correct generation of lower level $\mathbf{P}$-certificates in a tree-like fashion.

In our construction, we use $\mathsf{nuCerts}$ for $\mathbf{P}$ in place of the $\mathbf{P}$-certificates of [14]. We describe a sequence of protocols $\Pi_1, \Pi_2, \ldots$, where protocol Π_k uses k levels of $\mathsf{nuCerts}$ and allows us to simulate n^k concurrent sessions. Thus, to capture full concurrency, our final instantiation is Π_k for any $k \in \omega(1)$.

We proceed with the description of our protocol $\Pi_k = (P_k, V_k)$ for a language $L \in \mathbf{NP}$. We present the protocol assuming that Com is non-interactive, statistically binding commitment scheme, but this can be naturally replaced by any

constant-round commitment scheme (for example, the scheme of Naor [46,37] which consists of two rounds and relies on the existence of one-way functions). We further assume $\mathcal{H}_n = \{h\colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^n\}$ is a family of collision resistant hash functions mapping strings of arbitrary polynomial length to strings of length n. We can instantiate this using a Merkle tree based on any family of compressing collision resistant hash functions. Lastly, we assume $\mathsf{nuCerts}$ for $\mathbf{P}$ with length bounded by $\ell(n) \in \mathsf{poly}(n)$.

Let 1^n and x be common inputs to (P_k, V_k) and w be a private input to P_k such that $(x, w) \in \mathbf{R}_L$. The protocol proceeds in two phases. In the first phase, the prover commits to a message before receiving a "challenge" r from the verifier. In the second phase, the prover provides a WIUA proving either that x is in the $\mathbf{NP}$ language or that the message it committed to in phase 1 was actually a sequence of machines that outputs r. We summarize this protocol Π_k as follows:

Phase 1: P_k and V_k exchange the following three messages.

1. V_k chooses a randomly sampled hash function $h \xleftarrow{\$} \mathcal{H}_n$ and sends h to P_k.
2. P_k sends a commitment to 0^n using Com with randomness ρ.
3. V_k replies with a random "challenge" $r \xleftarrow{\$} \{0,1\}^{6\ell nk}$.

Phase 2: P_k provides a WIUA of the statement that either $x \in L$ OR there exists $\langle \vec{M}, j, \vec{s}, \vec{\pi}, \vec{\sigma}, \vec{\lambda}, \rho \rangle$ such that:

1. **Commitment Consistency:** c is a commitment to $h(\vec{M})$ using randomness ρ,
2. **Input Certification:**
   (a) $|\vec{\sigma}| \leq \ell nk$.
   (b) For $2 \leq i \leq k$, π_i certifies that $M_i(1^n, \zeta(j,i), s_i, ([\lambda_{\geq i}]_{\leq \zeta(j,i)}, [\sigma_{\geq i}]_{\leq \zeta(j,i)})) = \lambda_{i-1}$.
3. **Prediction Correctness:** π_1 certifies that $M_1(1^n, j, s_1, ([\lambda_{\geq 1}]_{\leq j}, [\sigma_{\geq 1}]_{\leq j})) = r$.

We define $\zeta(j,i) \triangleq j - (j \bmod n^{i-1})$, $\gamma_{\geq i} = (\gamma_i, \gamma_{i+1}, \ldots)$, and $[\gamma]_{\leq j} \triangleq \{(j', \cdot, \cdot) \in \gamma : j' \leq j\}$. These are used to "filter out" all unnecessary messages from future rounds.

There are two things we need to show: (1) there exists a simulator that can communicate with a cheating verifier in arbitrarily (yet polynomial) number of concurrent sessions and be able to convince it that the instance is in the language, even without having the witness, and (2) any *non-uniform* efficient cheating prover cannot convince the verifier the validity of a false statement. For (1), the ZK simulator is identical to [14] as $\mathsf{nuCerts}$ for $\mathbf{P}$ and $\mathbf{P}$-certificates have the same completeness and efficiency guarantees. We refer to the full version of the paper for the proof of (2) and hence Theorem 6.

References

1. Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS. pp. 106–115 (2001)
2. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM Journal on Computing **38**(5), 1661–1694 (2008)
3. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS. pp. 116–125 (2001)
4. Barak, B., Lindell, Y., Vadhan, S.P.: Lower bounds for non-black-box zero knowledge. J. Comput. Syst. Sci. **72**(2), 321–391 (2006)
5. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. **37**(2), 380–400 (2007)
6. Berman, I., Degwekar, A., Rothblum, R.D., Vasudevan, P.N.: Multi-collision resistant hash functions and their applications. In: Advances in Cryptology - EUROCRYPT. pp. 133–161 (2018)
7. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC. pp. 671–684 (2018)
8. Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: Theory of Cryptography - TCC. pp. 209–234 (2018)
9. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences **37**(2), 156–189 (1988)
10. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: 32nd Annual ACM Symposium on Theory of Computing, STOC. pp. 235–244 (2000)
11. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. SIAM J. Comput. **32**(1), 1–47 (2002)
12. Canetti, R., Lin, H., Paneth, O.: Public-coin concurrent zero-knowledge in the global hash model. In: Theory of Cryptography - TCC. pp. 80–99 (2013)
13. Chongchitmate, W., Ostrovsky, R., Visconti, I.: Resettably-sound resettable zero knowledge in constant rounds. In: Theory of Cryptography - TCC. pp. 111–138 (2017)
14. Chung, K., Lin, H., Pass, R.: Constant-round concurrent zero knowledge from p-certificates. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp. 50–59 (2013)
15. Chung, K., Lin, H., Pass, R.: Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In: Advances in Cryptology - CRYPTO. pp. 287–307 (2015)
16. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.P.: Random oracles and non-uniformity. In: Advances in Cryptology - EUROCRYPT. pp. 227–258 (2018)
17. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Advances in Cryptology - EUROCRYPT. pp. 418–430 (2000)
18. Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp. 251–260 (2009)
19. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. Journal of the ACM (JACM) **51**(6), 851–898 (2004)

20. Dwork, C., Sahai, A.: Concurrent zero-knowledge: Reducing the need for timing constraints. In: Advances in Cryptology - CRYPTO. pp. 442–457 (1998)
21. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 416–426. ACM (1990)
22. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO. pp. 186–194 (1986)
23. Fortnow, L.: Kolmogorov complexity and computational complexity. Complexity of Computations and Proofs. Quaderni di Matematica **13** (2004)
24. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO. pp. 297–315 (2011)
25. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: 41st Annual Symposium on Foundations of Computer Science, FOCS. pp. 305–313 (2000)
26. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: 43rd ACM Symposium on Theory of Computing, STOC. pp. 99–108 (2011)
27. Goldreich, O.: The Foundations of Cryptography - Volume 1, Basic Techniques. Cambridge University Press (2001)
28. Goldreich, O.: Concurrent zero-knowledge with timing, revisited. In: 34th Annual ACM Symposium on Theory of Computing, STOC. pp. 332–340 (2002)
29. Goldreich, O., Håstad, J.: On the complexity of interactive proofs with bounded communication. Inf. Process. Lett. **67**(4), 205–214 (1998)
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on computing **18**(1), 186–208 (1989)
31. Goyal, V.: Non-black-box simulation in the fully concurrent setting. In: Symposium on Theory of Computing Conference, STOC. pp. 221–230. ACM (2013)
32. Goyal, V., Jain, A., Ostrovsky, R., Richelson, S., Visconti, I.: Constant-round concurrent zero knowledge in the bounded player model. In: Advances in Cryptology - ASIACRYPT. pp. 21–40 (2013)
33. Gupta, D., Sahai, A.: On constant-round concurrent zero-knowledge from a knowledge assumption. In: Progress in Cryptology - INDOCRYPT. pp. 71–88 (2014)
34. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Trans. Information Theory **45**(6), 1757–1767 (1999)
35. Guruswami, V., Umans, C., Vadhan, S.P.: Unbalanced expanders and randomness extractors from parvaresh-vardy codes. J. ACM **56**(4), 20:1–20:34 (2009)
36. Haitner, I., Ishai, Y., Omri, E., Shaltiel, R.: Parallel hashing via list recoverability. In: Advances in Cryptology - CRYPTO. pp. 173–190 (2015)
37. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing **28**(4), 1364–1396 (1999)
38. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. pp. 723–732. ACM (1992)
39. Kilian, J.: Improved efficient arguments. In: Advances in Cryptology - CRYPTO. pp. 311–324. Springer (1995)
40. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In: 33rd Annual ACM Symposium on Theory of Computing, STOC. pp. 560–569 (2001)
41. Komargodski, I., Naor, M., Yogev, E.: White-box vs. black-box complexity of search problems: Ramsey and graph property testing. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS. pp. 622–632 (2017)

42. Komargodski, I., Naor, M., Yogev, E.: Collision resistant hashing for paranoids: Dealing with multiple collisions. In: Advances in Cryptology - EUROCRYPT. pp. 162–194. Springer (2018)
43. Komargodski, I., Yogev, E.: On distributional collision resistant hashing. In: Advances in Cryptology - CRYPTO. pp. 303–327 (2018)
44. Merkle, R.C.: A certified digital signature. In: Advances in Cryptology - CRYPTO. vol. 435, pp. 218–238. Springer (1989)
45. Micali, S.: Computationally sound proofs. SIAM Journal on Computing **30**(4), 1253–1298 (2000)
46. Naor, M.: Bit commitment using pseudorandomness. Journal of cryptology **4**(2), 151–158 (1991)
47. Pandey, O., Prabhakaran, M., Sahai, A.: Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In: Theory of Cryptography - TCC. pp. 638–667 (2015)
48. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Advances in Cryptology - EUROCRYPT. pp. 160–176 (2003)
49. Pass, R., Rosen, A., Tseng, W.D.: Public-coin parallel zero-knowledge for NP. J. Cryptology **26**(1), 1–10 (2013)
50. Pass, R., Tseng, W.D., Venkitasubramaniam, M.: Concurrent zero knowledge, revisited. J. Cryptology **27**(1), 45–66 (2014)
51. Pass, R., Tseng, W.D., Wikström, D.: On the composition of public-coin zero-knowledge protocols. SIAM J. Comput. **40**(6), 1529–1553 (2011)
52. Pass, R., Venkitasubramaniam, M.: On constant-round concurrent zero-knowledge. In: Canetti, R. (ed.) Theory of Cryptography - TCC. pp. 553–570 (2008)
53. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: 43rd Symposium on Foundations of Computer Science FOCS. pp. 366–375. IEEE Computer Society (2002)
54. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Advances in Cryptology - EUROCRYPT. pp. 415–431 (1999)
55. Unruh, D.: Random oracles and auxiliary input. In: Advances in Cryptology - CRYPTO. pp. 205–223 (2007)