Constant-Round Concurrent Zero-knowledge from Indistinguishability Obfuscation

Kai-Min Chung*   Huijia Lin†   Rafael Pass‡

December 18, 2014

Abstract

We present a constant-round concurrent zero-knowledge protocol for NP. Our protocol relies on the existence of families of collision-resistant hash functions, one-way permutations, and indistinguishability obfuscators for $\mathbf{P}/poly$ (with slightly super-polynomial security).

*Academia Sinica, kmchung@iis.sinica.edu.tw

†University of California, Santa Barbara, rachel.lin@cs.ucsb.edu.

‡Cornell University, rafael@cs.cornell.edu. Work supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

1 Introduction

Zero-knowledge ($\mathcal{ZK}$) interactive proofs [GMR89] are paradoxical constructs that allow one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the Verifier. Beyond being fascinating in their own right, $\mathcal{ZK}$ proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks.

The notion of concurrent zero knowledge, first introduced and achieved in the paper by Dwork, Naor and Sahai [DNS04], considers the execution of zero-knowledge proofs in an asynchronous and concurrent setting. More precisely, we consider a single adversary mounting a coordinated attack by acting as a verifier in many concurrent executions (called sessions). Concurrent $\mathcal{ZK}$ proofs are significantly harder to construct and analyze. Since the original protocol by Dwork, Naor and Sahai (which relied on so called "timing assumptions"), various other concurrent $\mathcal{ZK}$ protocols have been obtained based on different set-up assumptions (e.g., [DS98, Dam00, CGGM00, Gol02, PTV12, GJO$^+$12]), or in alternative models (e.g., super-polynomial-time simulation [Pas03, PV10]).

In the standard model, without set-up assumptions (the focus of our work,) Canetti, Kilian, Petrank and Rosen [CKPR01] (building on earlier works by [KPR98, Ros00]) show that concurrent $\mathcal{ZK}$ proofs for non-trivial languages, with "black-box" simulators, require at least $\tilde{\Omega}(\log n)$ number of communication rounds. Richardson and Kilian [RK99] constructed the first concurrent $\mathcal{ZK}$ argument in the standard model without any extra set-up assumptions. Their protocol, which uses a black-box simulator, requires $O(n^\epsilon)$ number of rounds. The round-complexity was later improved in the work of Kilian and Petrank (KP) [KP01] to $\tilde{O}(\log^2 n)$ round. More recent work by Prabhakaran, Rosen and Sahai [PRS02] improves the analysis of the KP simulator, achieving an essentially optimal, w.r.t. black-box simulation, round-complexity of $\tilde{O}(\log n)$; see also [PTV12] for an (arguably) simplified and generalized analysis.

The central open problem in the area is whether a *constant-round* concurrent $\mathcal{ZK}$ protocol (for a non-trivial language) can be obtained. Note that it could very well be the case that all "classic" zero-knowledge protocols already are concurrent zero-knowledge; thus, simply assuming that those protocols are concurrent zero-knowledge yields an assumption under which constant-round concurrent zero-knowledge (trivially) exists—in essence, we are assuming that for every attacker a simulator exists. Furthermore, as shown in [GS12] (and informally discussed in [CLP13b]) under various "extractability" assumptions of the knowledge-of-exponent type [Dam91, HT98, BP04], constant-round concurrent zero-knowledge is easy to construct. But such extractability assumptions also simply assume that for every attacker, a simulator (in essence, "the extractor" guaranteed by the extractability assumption) exists. In particular, an explicit construction of the concurrent zero-knowledge simulator is not provided—it is simply assumed that one exists. For some applications of zero-knowledge such as *deniability* (see e.g., [DNS04, Pas03]), having an explicit simulator is crucial. Rather, we are here concerned with the question of whether constant-round concurrent zero-knowledge, *with an explicit simulator* exits.

1.1 Towards Constant-round Concurrent Zero-Knowledge

Recently, the authors [CLP13b] provided a first construction a constant-round concurrent zero-knowledge protocol with an explicit simulator, based on a new cryptographic hardness assumption—the existence of so-called **P**-*certificates*, roughly speaking, succinct non-interactive arguments for languages in **P**. An issue with their approach, however, is we only have candidate constructions of **P**-certificates that are sound against *uniform* polynomial-time attackers (as opposed to non-uniform ones), and the protocol of [CLP13b] inherits the soundness property of the underlying **P**-certificate.

Additionally, whereas the assumption that a particular proof system is a **P**-certificates is a falsifiable assumption [Pop63, Nao03], it is unclear whether the existence of **P**-certificates itself can be based on some more natural hardness assumption.

A very recent elegant work by Pandey, Prabhakaran and Sahai [PPS13] takes a different approach and instead demonstrates the existence of constant-round concurrent zero-knowledge protocol with an explicit simulator based on the existence of *differing-input obfuscation (**diO**)* for (restricted classes of) **P**/*poly* [BGI$^+$01, BCP14, ABG$^+$13]. Whereas the assumption that a particular scheme is a **diO** is an "extractability" assumption (similar in flavor to knowledge-of-exponent type [Dam91, HT98, BP04] assumptions), the intriguing part of the scheme of Pandey et al [PPS13] is that the extractability assumption is only used to prove *soundness* of the protocol; concurrent zero-knowledge is proved in the "standard" model, through providing an explicit simulator. Nevertheless, **diO** is a strong and subtle assumption—as shown by recent work [BP13, GGHW13, IPS14], unless we restricting the class of programs for which **diO** should hold, we may end up with a notion that is unsatisfiable. Additionally, there are currently no known approaches for basing **diO** on more "natural" (or in fact *any*) hardness (as opposed to extractability) assumption.

1.2 Our Results

In this paper, we combine the above-mentioned two approaches. Very roughly speaking, we will use obfuscation to obtain a variant of the notion of a **P**-certificate, and we next show that this variant still suffices to obtain constant-round concurrent zero-knowledge (where the soundness conditions holds also against non-uniform PPT attackers). More importantly, rather than using **diO**, we are able to use *indistinguishability obfuscation (**iO**)* [BGI$^+$01, GGH$^+$13]. Following the groundbreaking work of Garg et al [GGH$^+$13], there are now several candidate constructions of **iO** that can be based on hardness assumptions on (approximate) multilinear maps [PST14, GLSW14].

Theorem. *Assume the existence of indistinguishability obfuscation for* **P**/*poly (with slightly super-polynomial security), one-way permutations (with slightly super-polynomial security) and collision-resistant hash function. Then there exists a constant-round concurrent zero-knowledge argument for* NP.

In more details, our approach proceeds in the following steps:

- We first observe that a warm-up case considered in [CLP13b]—which shows the existence of constant-round concurrent zero-knowledge based on, so-called, *unique* **P***-certificates* (that is, **P**-certificates for which there exists at most one accepting certificate for each statement) directly generalizes also to unique **P**-certificates in the Common *Random* String model (a.k.a. the Uniform Random String model (URS)) satisfying an *adaptive soundness* property (where the statement to be proved can be selected after the URS).

- We next show that by appropriately modifying the protocol, we can handle also unique **P**-certificates in the URS model satisfying even just a "static" soundness condition (where the statement needs to be selected before the URS is picked), and additionally also unique **P**-certificates (with static soundness) in the Common Reference String (CRS) model, where the reference string no longer is required to be uniform. Unique **P**-certificates in the CRS model (also with non-uniform soundness) can be constructed based on the existence of **diO** for (a restricted class of) **P**/*poly* [BP13], and as such this preliminary step already implies the result of [PPS13] in a modular way (but with worse concrete round complexity).

- We next consider a more relaxed variant of unique **P**-certificates in the CRS model—which we refer to as *delegatable unique* ***P**-certificates*—where the CRS is allowed to be *statement dependent* but only a "small" (in particular, independent of the statement length) part of the CRS generation requires using secret coins. By relying on **iO** for **P**/*poly*, we next show that the protocol can be generalized to work also with such delegatable unique **P**-certificates.

- We finally leverage recent results on delegation of computation based on **iO** from [BGL$^+$14, CHJV14, KLW14] and show that the beautiful protocol of Koppula, Lewko and Waters [KLW14] can be modified into a delegatable unique **P**-certificate (also with non-uniform soundness).

1.3 Outline of Our Techniques

We here provide a detailed outline of our techniques. As mentioned, our construction heavily relies on a "warm-up" case of the construction of [CLP13b], which we start by recalling (closely following the description in [CLP13b]). The starting point of the construction of [CLP13b] is the construction is Barak's [Bar01] non-black-box zero-knowledge argument for NP. We start by very briefly recalling the ideas behind his protocol (following a slight variant of this protocol due to [PR03b]).

Barak's protocol Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\text{poly}(n)}$, the Prover **P** and Verifier V, proceed in two stages. In Stage 1, P starts by sending a computationally-binding commitment $c \in \{0,1\}^n$ to 0^n; V next sends a "challenge" $r \in \{0,1\}^{2n}$. In Stage 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 $M(\sigma) = r$.[1]

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 $\sigma \in \{0,1\}^n$. 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 actually be a witness indistinguishable *universal argument* (WIUA) [Mic00, BG08] since the statement that c is a commitment to a program M of *arbitrary* polynomial-size, and that $M(c) = r$ within some *arbitrary* polynomial time, is not in NP.

Now, let us consider concurrent composition. That is, we need to simulate the view of a verifier that starts *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. (Note that if we increase the length of r we can handle a bounded number of concurrent executions, by simply letting σ include all these messages).

So, if the simulator could commit not only to the code of V^*, but also to a program M that generates all other prover messages, then we would seemingly be done. And at first sight, this doesn't seem impossible: since the simulator S is actually the one generating all the prover messages, why don't we just let M be an appropriate combination of S and V^*? This idea can indeed be implemented [PR03b, PRT11], but there is a serious issue: if the verifier "nests" its concurrent

[1] We require that C is a commitment scheme allowing the committer to commit to an arbitrarily long string $m \in \{0,1\}^*$. Any commitment scheme for fixed-length messages can easily be modified to handle arbitrarily long messages by asking the committer to first hash down m using a collision-resistant hash function h chosen by the receiver, and next commit to $h(m)$.

executions, the running-time of the simulation quickly blows up exponentially—for instance, if we have three nested sessions, to simulate session 3 the simulator needs to generate a WIUA regarding the computation needed to generate a WIUA for session 2 which in turn is regarding the generation of the WIUA of session 1 (so even if there is just a constant overhead in generating a WIUA, we can handle at most $\log n$ nested sessions).

Unique P-certificates to The Rescue: The "Warm-Up" Case from [CLP13b] As shown in [CLP13b], the blow-up in the running-time can be prevented using Unique **P**-certificates. Roughly speaking, we say that (P, V) is a **P**-*certificate system* if (P, V) is a non-interactive proof system (i.e., the prover send a single message to the verifier, who either accepts or rejects) allowing an efficient prover to convince the verifier of the validity of any *deterministic polynomial-time computation* $M(x) = y$ using a "certificate" of some *fixed* polynomial length (independent of the size and the running-time of M) whose validity the verifier can check in some fixed polynomial time (independent of the running-time of M). The **P**-certificate system is *unique* if there exists at most one accepted proof for any statement.

The protocol proceeds just as Barak's protocol except that Stage 2 is modified as follows: instead of having P prove (using a WIUA) that either x is true, or there exists a "short" string $\sigma \in \{0, 1\}^{2n}$ such that c is a commitment to a program M such that $M(\sigma) = r$, we now ask P to use a WIUA to prove that either x is true, or

- **commitment consistency:** c is a commitment to a program M_1, and
  - **input certification:** there exists a vector $\lambda = ((1, \pi_1), (2, \pi_2), \ldots)$ and a vector of messages $\vec{m}$ such that π_j certifies that $M_1(\lambda_{<j})$ outputs m_j in its j'th communication round, where $\lambda_{<j} = ((1, \pi_1), \ldots, (j-1, \pi_{j-1}))$, and
  - **prediction correctness:** there exists a **P**-certificate π of length n demonstrating that $M_1(\lambda) = r$.

Soundness of the modified protocol, roughly speaking, follows since by the unique certificate property, for every program M_1 it inductively follows that for every j, m_j is uniquely defined, and thus also the *unique* (accepting) certificate π_j certifying $M_1(\lambda_{<j}) = m_j$; it follows that M_1 determines a unique vector λ that passes the input certification conditions, and thus there exists a single r that make M_1 also pass the prediction correctness conditions. Note that we here inherently rely on the fact that the **P**-certificate is unique to argue that the sequence λ is uniquely defined. (Technically, we here need to rely on a **P**-certificate that is sound for slightly super-polynomial-time as there is no a-priori polynomial bound on the running-time of M_1, nor the length of λ.)

To prove zero-knowledge, roughly speaking, our simulator will attempt to commit to its own code in a way that prevents a blow-up in the running-time. Recall that the main reason that we had a blow-up in the running-time of the simulator was that the generation of the WIUA is expensive. Observe that in the new protocol, the only expensive part of the generation of the WIUA is the generation of the **P**-certificates π; the rest of the computation has *a-priori* bounded complexity (depending only on the size and running-time of V^*). To take advantage of this observation, we thus have the simulator only commit to a program that generates prover messages (in identically the same way as the actual simulator), but getting certificates $\vec{\pi}$ as input.

In more detail, to describe the actual simulator S, let us first describe two "helper" simulators S_1, S_2. S_1 is an interactive machine that simulates prover messages in a "right" interaction with V^*. Additionally, S_1 is expecting some "external" messages on the "left"—looking forward, these

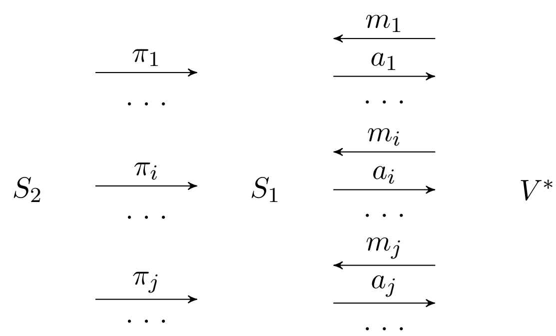

Figure 1: Simulation using **P**-certificates.

"left" messages will later be certificates provided by S_2. See Figure 1 for an illustration of the communication patterns between S_1, S_2 and V^*.

S_1 proceeds as follows in the right interaction. In Stage 1 of every session i, S_1 first commits to a machine $\tilde{S}_1(j', \tau)$ that emulates an interaction between S_1 and V^*, feeding S_1 input τ as messages on the left, and finally $\tilde{S}_1$ outputs the verifier message in the j''th communication round in the right interaction with V^*. (Formalizing what it means for S_1 to commit to $\tilde{S}_1$ is not entirely trivial since the definition of $\tilde{S}_1$ depends on S_1; we refer the reader to the formal proof for a description of how this circularity is broken.[2] S_1 next simulates Stage 2 by checking if it has received a message (j, π_j) in the left interaction, where j is the communication round (in the right interaction with V^*) where the verifier sends its random challenge and expects to receive the first message of Stage 2; if so, it uses $M_1 = \tilde{S}_1$ (and the randomness it used to commit to it), j and σ being the list of messages received by S_1 in the left interaction, as a "fake" witness to complete Stage 2.

The job of S_2 is to provide **P**-certificates π_j for S_1 allowing S_1 to complete its simulation. S_2 emulates the interaction between S_1 and V^*, and additionally, at each communication round j, S_2 feeds S_1 a message (j, π_j) where π_j is a **P**-certificate showing that $\tilde{S}_1(j, \sigma_{<j}) = r_j$, where $\sigma_{<j}$ is the list of messages already generated by S_2, and r_j is the verifier message in the j'th communication round. Finally, S_2 outputs its view of the full interaction.

The actual simulator S just runs S_2 and recovers from the view of S_2 the view of V^* and outputs it. Note that since S_1 has polynomial running-time, generating each certificate about $\tilde{S}_1$ (which is just about an interaction between S_1 and V^*) also takes polynomial time. As such S_2 can also be implemented in polynomial time and thus also S.

Finally, indistinguishability of this simulation, roughly speaking, follow from the hiding property of the commitment in Stage 1, and the WI property of the WIUA in Stage 2. (There is another circularity issue that arises in formalizing this—as S_1 in essence needs to commit to its own randomness—but it can be dealt with as shown in [CLP13b]; in this overview, we omit the details as they are not important for our modifications to the protocol, but they can be found in the formal proof.)

Generalizing to Unique P-certificates in CRS model The key technical contribution in [CLP13b] was to generalize the above approach to deal also with "non-unique" **P**-certificates. Here we instead aim to generalize the above approach to work with **P**-certificates in the CRS model, but still relying on the uniqueness property.

[2] Roughly speaking, we let S_1 take the description of a machine M as input, and we then run S_1 on input $M = S_1$.

Let us first note that if we had access to unique **P**-certificate in the URS (i.e., the uniform reference string) model satisfying an *adaptive soundness* property (where the statement to be proved can be selected after the URS, then above-mentioned protocol almost directly generalized to work with them (as opposed to using unique **P**-certificates in the "plain" model) by simply having the Verifier send the URS ρ along with its first message of the protocol.[3] The only issue that needs to be addressed in implementing this change is to specify what it means that "π_j certifies that $M_1(\lambda_{<j})$ outputs m_j" in the input certification step in Stage 2, since this certification needs to be done with respect to some URS. We modify Stage two to require that M_1 outputs not only messages m_i, but also reference strings ρ_i. Let us remark that to ensure that soundness still holds, we require the **P**-certificate system to satisfy a strong uniqueness property: uniqueness of accepting proofs needs to hold for *all* reference strings ρ.

We next note that the protocol can be further generalized to handle also unique **P**-certificates in the URS model satisfying even just a *static soundness* condition (where the statement needs to be selected before the URS is picked) by proceeding as follows:

- We add a Stage 1.5 to the protocol where the Prover is asked to provide a commitment c_2 to 0^n and then asked to provide a WIUARG that either $x \in L$ or c_2 is a commitment to a "well-formed" statement (but not that the statement is true) for the **P**-certificate in use in Stage 2.

- Stage 2 of the protocol is then modified to first have the Verifier send the URS for the **P**-certificate, and then requiring that the prover uses a **P**-certificate for the statement committed to in c_2. In other words, we require the Prover to commit in advance, and prove knowledge of, the statement to be used in the **P**-certificate and thus static soundness suffices.

Additionally, this approach generalizes also to deal with unique **P**-certificates in the Common *Reference* String (CRS) model (where the reference string no longer needs to be uniform), by having the Verifier provide a zero-knowledge proof that the CRS was well-formed.[4] Let us again remark that to ensure that soundness still holds, we require the uniqueness property of the **P**-certificate system to hold for all reference strings ρ, *even invalid ones*.

Generalizing to Delegatable P-certificates The notion of a **P**-certificate in the CRS model requires that the same CRS can be used to prove *any* statement q of any (polynomially-related) length. We will now consider a weaker notion of a **P**-certificate in the CRS model, where the CRS is "statement-dependent"—that is, the CRS is generated as a function of the statement q to be proved—in essence, such **P**-certificates can be viewed as specific instances of a *two-round* delegation protocol. But whereas the CRS may depend on the statement, we still restrict it in several important ways:

- As before, the length of the CRS is "short" (independent of the length of the statement q).

- Additionally, only a "small" part of the generation procedure relies on secret coins. More precisely, the CRS generation procedure proceeds in three steps: 1) first, secret coins are used

[3]To make this work, we need to rely on **P**-certificates in the URS model with perfect completeness. This requirement can be removed by additionally performing a coin-tossing to determine the URS. For simplicity of exposition, we here simply assume perfect completeness.

[4]Again, we here rely on **P**-certificates in the CRS model with perfect completeness. This requirement can also be avoided by having the prover and the verifier perform coin-tossing-in-the-well to determine the secret coins the verifier should use for generating the CRS. As our instantiations of **P**-certificates will satisfy perfect completeness, we do not further formalize this approach.

to generate a public parameter PP and a secret parameter K (this is done independently of the statement q), 2) next, only PP is used to *deterministically* process the statement q into a "short" digest d (independent of the length of q), and 3) the digest d and the secret parameter K is efficiently processed to finally generate the CRS (independent of the length of q). To emphasize, only step 2 requires work that is proportional to the length of q, but this work only requires public information.

We now generalize the above approach to also work with delegatable unique **P**-certificates.

- Instead of having the Verifier send the CRS in the clear (which it cannot compute as it does not know the statement q on which it will be run), it simply runs part 1 of the CRS generation procedure to generate PP and K and sends just the public-parameter PP to the Prover.

- The Prover is then asked to provide a third commitment c_3 to 0^n and provide a WIUARG that either $x \in L$ or c_3 is a correctly computed digest d (w.r.t., PP) to the statement q committed to in c_2. (In essence, the Verifier is delegating the computation of d to the Prover.)

- Next, the Verifier sends an indistinguishability obfuscation $\tilde{\Pi} = \mathbf{iO}(\Pi)$ of a program Π that on input a decommitment (d', r') to c_3 processes d' and K into a CRS ρ and outputs it. (The reason that the Verifier cannot generate ρ in the clear is that digest d cannot be sent to the Verifier in the clear; recall that the honest prover will never compute any such digest, it is meant to commit to 0^n and prove that $x \in L$.) Additionally, the verifier gives a zero-knowledge proof that the obfuscation is correctly computed (and using the same random coins that were used to generate PP).

- Then, the Prover provides a commitment c_4 to 0^n and provides a WI proof of knowledge that $x \in L$ or c_4 is a commitment to a CRS ρ computed by applying the obfuscated code $\tilde{\Pi}$ to a proper decommitment of c_3.

- Finally, in Stage 2 of the protocol, we require the Prover to provide **P**-certificates w.r.t to the CRS ρ committed to in c_4.

Note that if c_3 is perfectly binding, then by **iO** security of the obfuscation, we can replace Π with a program that has the CRS ρ hardcoded and does not depend on K, and this suffices for arguing that soundness of the protocol still holds. On the other hand, the simulation can proceed just as before except that the simulator now uses the obfuscated code $\tilde{\Pi}$ to generate the CRS ρ and commits to it in c_4.

This completes the informal description of our protocol and its proof of security. In the formal description of the protocol in Section 4, we directly provide a construction using delegatable unique **P**-certificates, without going through the intermediary, simpler, cases mentioned above. As mentioned above, the above description ignores certain subtleties required to prevent circularities in the simulation and the proof of security. To deal with these issue (already considered in [CLP13b]) as well as to streamline the description of the final protocol (to enable a better concrete round-complexity) the formal description slightly difference from what is outlined above.

Realizing Delegatable Unique P-Certificates We finally leverage recent results on delegation of computation based on **iO** for circuits from [BGL$^+$14, CHJV14, KLW14] and show that the beautiful protocol of Koppula, Lewko and Waters [KLW14] can be massaged (and slightly modified) into a delegatable unique **P**-certificate.

Let is point out that, just as [CLP13b], our protocol requires the use of **P**-certificates that satisfy a slightly strong soundness condition—namely, we require soundness to hold against circuits of size $T(\cdot)$ where $T(\cdot)$ is some "nice" (slightly) super-polynomial function (e.g., $T(n) = n^{\log\log\log n}$). To achieve such (delegatable) **P**-certificates, we thus need to rely on **iO** for $\mathbf{P}/poly$ secure against $T(\cdot)$-size circuits.

1.4 Other Related Work

Since the work of Barak [Bar01], non-black-box simulation techniques have been used in several other contexts: For example, non-malleability [Bar02, Pas04, PR05a, PR05b], resettable-soundness [BGGL01, DGS09, BP12, CPS13, COPV13, COP$^+$14], concurrent secure computation [Lin03, PR03a, Pas04, BS05], covert secure computation [GJ10] and more. We believe our techniques may yield improved constructions also in these settings.

We also mention recent work of [CLP13a, Goy13] that constructs *public-coin* concurrent zero-knowledge protocols using non-black-box simulation; these protocols are not constant-round but instead rely on "standard" assumptions. Let us finally mention that the constant-round concurrent zero-knowledge protocol of [CLP13b] (which relies on non-interactive **P**-certificates) actually also is public-coin, whereas our protocol is not. We leave open the question of basing public-coin concurrent zero-knowledge on **iO**.

2 Preliminaries

Let $\mathcal{N}$ denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \ldots, n\}$. We denote by PPT probabilistic polynomial time Turing machines. We assume familiarity with interactive Turing machines, denoted ITM, 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 with private input x, when the random tape of each machine is uniformly and independently chosen, and $\mathsf{View}_B \langle A(x), B(y)\rangle(z)$ the random variable representing B's view in such an interaction. The term *negligible* is used for denoting functions that are (asymptotically) smaller than one over any polynomial. More precisely, a function $\nu(\cdot)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n, it holds that $\nu(n) < n^{-c}$.

2.1 Statistically Binding Commitment Schemes

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**. Commitment schemes come in two different flavors, statistically (or perfectly) binding and statistically (or perfectly) hiding; we only make use of statistically (and perfectly) binding commitments in this paper. Below we sketch the properties of a statistically (and perfectly) binding commitment; full definitions can be found in [Gol01].

In statistically (perfectly) binding commitments, the binding property holds against unbounded adversaries, while the hiding property only holds against computationally bounded (non-uniform) adversaries. The statistically (perfectly) binding property asserts that, with overwhelming probability (or probability 1) over the randomness of the receiver, the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that the commitments to any two different values are computationally indistinguishable.

Non-interactive perfectly-binding commitment schemes can be constructed using any one-to-one one-way function (see Section 4.4.1 of [Gol01]). Two-message statistically binding commitment schemes, in which the receiver first sends a single random initialization message, can be obtained from any one-way function [Nao91, HILL99].

2.2 Interactive Proofs and Arguments

We recall the standard definitions of interactive proofs [GMR89] and arguments (a.k.a computationally sound proofs) [BCC88].

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* $n \in N$, $x \in L$, *and every* $w \in R_L(x)$,

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

- Soundness: *For every pair of machines* B_1, B_2 *and every* $n \in N$,

$$\Pr[(x, z) \leftarrow B_1(1^n) : x \notin L \wedge (B_2(z), V)(1^n, x) = 1] \leq \nu(n)$$

If the soundness condition only holds against all non-uniform polynomial-time machines B_1, B_2, *the pair* (P, V) *is called an* **interactive argument system***.*

2.3 Witness Indistinguishability

An interactive protocol is **witness indistinguishable** (WI) [FS90] 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* $L \in \mathsf{NP}$ *is* **witness indistinguishable** *for* R_L *if for every PPT adversarial verifier* V^*, *and for every two sequences* $\{w^1_{n,x}\}_{n\in N, x\in L\cap\{0,1\}^{\mathrm{poly}(n)}}$ *and* $\{w^2_{n,x}\}_{n\in N, x\in L\cap\{0,1\}^{\mathrm{poly}(n)}}$, *such that* $w^1_{n,x}, w^2_{n,x} \in R_L(x)$ *for every* $n \in N$ *and* $x \in L \cap \{0,1\}^{\mathrm{poly}(n)}$, *the following ensembles are computationally indistinguishable over* N*:*

- $\{\mathsf{View}_{V^*}\left\langle P(w^1_{n,x}), V^*(z)\right\rangle(1^n, x)\}_{n\in N, x\in L\cap\{0,1\}^{poly(n)}, z\in\{0,1\}^*}$
- $\{\mathsf{View}_{V^*}\left\langle P(w^2_{n,x}), V^*(z)\right\rangle(1^n, x)\}_{n\in N, x\in L\cap\{0,1\}^{poly(n)}, z\in\{0,1\}^*}$

2.4 Special-sound WI proofs

A 4-round public-coin interactive proof for the language $L \in \mathcal{NP}$ with witness relation R_L is special-sound with respect to R_L, if for any two transcripts $(\delta, \alpha, \beta, \gamma)$ and $(\delta', \alpha', \beta', \gamma')$ such that the initial two messages, δ, δ' and α, α', are the same but the challenges β, β' are different, there is a deterministic procedure to extract the witness from the two transcripts and runs in polynomial time. In this paper, we rely on special sound proofs that are also witness indistinguishable. Special-sound WI proofs ($\mathcal{WISSP}$ for short) for languages in NP can be based on the existence of 2-round commitment schemes, which in turn can be based on one-way functions [GMW91, FS90, HILL99, Nao91].

2.5 Universal Arguments

Universal arguments (introduced in [BG08] and closely related to the notion of CS-proofs [Mic00]) 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 [BG08, Mic00]. A triplet $y = (M, x, t) \in L_{\mathcal{U}}$ 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 x 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. Below we recall the definition in [BG08].

Definition 3 (Universal argument)**.** *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 such that for any $y = (M, x, t)$, the total time spent by the (probabilistic) verifier strategy V, on common input 1^n, y, is at most $p(n + |y|)$. In particular, all messages exchanged in the protocol have length smaller than $p(n + |y|)$.*

- Completeness by a relatively efficient prover: *For every $n \in N$, $y = (M, x, t) \in L_{\mathcal{U}}$ and w in $\mathbf{R}_{\mathcal{U}}(y)$,*

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

Furthermore, there exists a polynomial q such that the total time spent by $P(w)$, on common inputs 1^n and (M, x, t), is at most $q(n + |y| + T_M(x, w)) \leq q(n + |y| + t)$.

- Computational Soundness: *For every polynomial size circuit family $\{P_n^*\}_{n \in N}$, there is a negligible function ν, such that, for every $n \in N$ and every triplet $(M, x, t) \in \{0,1\}^{\text{poly}(n)} \setminus L_{\mathcal{U}}$,*

$$\Pr[(P_n^*, V)(1^n, (M, x, t)) = 1] < \nu(n)$$

- Weak proof of knowledge: *For every positive polynomial p there exists a positive polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds: for every polynomial-size circuit family $\{P_n^*\}_{n \in N}$, every sufficiently large $n \in N$ and every $y = (M, x, t) \in \{0,1\}^{\text{poly}(n)}$ if $\Pr[(P_n^*, V)(1^n, y) = 1] > 1/p(n)$ then*

$$\Pr_r[\exists w = w_1, \ldots w_t \in \mathbf{R}_{\mathcal{U}}(y) \;\; s.t. \;\; \forall i \in [t], \; E_r^{P_n^*}(1^n, y, i) = w_i] > \frac{1}{p'(n)}$$

where $\mathbf{R}_{\mathcal{U}}(y) \stackrel{\text{def}}{=} \{w : (y, w) \in \mathbf{R}_{\mathcal{U}}\}$ and $E_r^{P_n^*}(\cdot, \cdot, \cdot)$ denotes the function defined by fixing the random-tape of E to equal r, and providing the resulting E_r with oracle access to P_n^*.

The weak proof-of-knowledge property of universal arguments only guarantees that each individual bit w_i of some witness w can be extracted in probabilistic polynomial time. Given an input 1^n and $y = (M, x, t)$ in $L_{\mathcal{U}} \cap \{0,1\}^{\text{poly}(n)}$, since the witness $w \in \mathbf{R}_{\mathcal{U}}(y)$ is of length at most t, it follows that there exists a extractor running in time polynomial in $\text{poly}(n) \cdot t$ that extracts the whole witness; we refer to this as the *global proof-of-knowledge property* of a universal argument.

The notion of witness indistinguishability of universal argument for $\mathbf{R}_{\mathcal{U}}$ is defined similarly as that for interactive proofs/arguments for NP relations; we refer the reader to [BG08] for a formal definition. [BG08] (based on [Mic00, Kil95]) presents a witness indistinguishable universal argument (WIUA for short) based on the existence of families of collision-resistant hash functions.

2.6 Concurrent Zero-Knowledge

An interactive proof is said to be **zero-knowledge** if it yields nothing beyond the validity of the statement being proved [GMR89].

Definition 4 (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 $n \in N$:*

- $\{\mathsf{View}_{V^*}\langle P(w), V^*(z)\rangle(1^n, x)\}_{n\in N, x\in L\cap\{0,1\}^{poly(n)}, w\in R_L(x), z\in\{0,1\}^{\mathrm{poly}(n)}}$
- $\{S(1^n, x, z)\}_{n\in N, x\in L\cap\{0,1\}^{poly(n)}, w\in R_L(x), z\in\{0,1\}^{\mathrm{poly}(n)}}$

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 $\mathsf{View}_{V^*}\langle P, V^*(z)\rangle(1^n, x)$ 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 w witness of x.

Definition 5 (Concurrent Zero-Knowledge [DNS04]). *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 following two ensembles are computationally indistinguishable over $n \in N$.*

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

2.7 Forward Secure PRG

We recall the definition of forward secure PRG from [CLP13b]. Roughly speaking, a *forward-secure pseudorandom generator (PRG)* (first formalized by [BY03], but early usages go back to [BH92]) is a pseudorandom generator where the seed is periodically updated—thus we have a sequence of seeds $s_1, s_2, \ldots$ generating a pseudorandom sequence $q_1, q_2, \ldots$—such that if the seed s_t is exposed (and thus the "later" sequence $q_{t+1}, q_{t+2}, \ldots$ is also exposed), the "earlier" sequence $q_1, \ldots, q_t$ still remains pseudorandom.

We provide a simple definition of a forward secure pseudorandom generator, where the "exposure" time t is statically selected.[5]

Definition 6 (Forward-secure Pseudorandom Generator). *We say that a polynomial-time computable function G is a* forward secure Pseudo-Random Generator (fsPRG) *if on input a string s, and $\ell \in N$, it outputs two sequences $(s_1, s_2, \ldots s_\ell)$ and $(q_1, q_2, \ldots, q_\ell)$ such that the following properties hold:*

- Consistency: *For every $n, \ell \in N$, $s \in \{0,1\}^n$, the following holds*

[5]The definition of [BY03] allows an attacker to adaptively select the exposure time t. For our purposes the simpler non-adaptive notion suffices.

– *if* $G(s,\ell) = ((s_1, \vec{s}), (q_1, \vec{q}))$*, then* $G(s_1, \ell-1) = (\vec{s}, \vec{q})$*.*

- Forward Security: *For every polynomial* p*, the following ensembles are computationally indistinguishable*

  – $\{s \leftarrow U_n, (\vec{s}, \vec{q}) \leftarrow G(s,\ell) : s_t, \vec{q}_{\leq t}\}_{n \in N, \ell \in [p(n)], t \in [\ell]}$

  – $\{s_t \leftarrow U_n, \vec{q} \leftarrow (U_n)^\ell : s_t, \vec{q}_{\leq t}\}_{n \in N, \ell \in [p(n)], t \in [\ell]}$

  where U_n *is the uniform distribution over* $\{0,1\}^n$ *and* $\vec{q}_{\leq t} = (q_1, \ldots, q_t)$*.*

Any (traditional) PRG implies the existence of a forward secure PRG; thus by the result of [HILL99] the existence of forward secure PRGs are implied by the existence of one-way functions.

In our application of forward secure PRGs, we will use the outputs of the PRG in *reverse order*, and thus write $G(s,\ell) = (s_\ell, s_{\ell-1}, \ldots s_1), (q_\ell, q_{\ell-1}, \ldots, q_1)$. As a consequence, we may reveal a seed s_t "explaining" the "earlier" sequence $((s_{t-1}, \ldots s_1), (q_{t-1}, \ldots, q_1))$ while guaranteeing that the "later" sequence $(q_\ell, \ldots q_t)$ still is indistinguishable from random.

2.8 Indistinguishability Obfuscation

We recall the definition of indistinguishability obfuscation for polynomial-sized circuits of [BGI+01].

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$))**.** *A uniform machine* $i\mathcal{O}$ *is a indistinguishability obfuscator for a class of* deterministic *circuits* $\{\mathcal{C}_k\}_{k\in\mathbb{N}}$*, if the following conditions are satisfied:*

Correctness: *For all security parameters* $k \in \mathbb{N}$*, for all* $C \in \mathcal{C}_k$*, for all input* x*, we have that*

$$\Pr[\Lambda \leftarrow i\mathcal{O}(1^k, C) \ : \ \Lambda(x) = C(x)] = 1$$

Security: *For every* non-uniform PPT *samplable distribution* $\mathcal{D}$ *over the support* $\{\mathcal{C}_k \times \mathcal{C}_k \times \{0,1\}^{\mathrm{poly}(k)}\}$*, and adversary* A^**, there is a negligible function* μ*, such that, for sufficiently large* $k \in \mathbb{N}$*, if*

$$\Pr[(C_1, C_2, z) \leftarrow \mathcal{D}(1^k) \ : \ \forall x, \ C_1(x) = C_2(x)] > 1 - \mu(k)$$

Then, the following holds

$$\begin{aligned} &|\Pr[(C_1, C_2, z) \xleftarrow{\$} \mathcal{D}(1^k) \ : \ A^*(i\mathcal{O}(1^k, C_1), z)] \\ &- \Pr[(C_1, C_2, z) \xleftarrow{\$} \mathcal{D}(1^k) \ : \ A^*(i\mathcal{O}(1^k, C_2), z)]| \quad \leq \mu(k) \end{aligned}$$

Furthermore, we say that $i\mathcal{O}$ *is* **super-polynomially secure** *if there is a super-polynomial function* T*, such that, the above condition holds for all* T*-time adversary* A^**.*

Definition 2. *A uniform* PPT *machine* $i\mathcal{O}(\cdot,\cdot)$ *is an indistinguishability obfuscator for polynomial-sized circuits if it is an indistinguishability obfuscator for the class of circuits* $\{\mathcal{C}_k\}_{k\in N}$ *containing all circuits of size at most* k*.*

3 P-certificates

We first define **P**-certificates in the CRS model where the CRS is independent of the statement being proved. Then, we move to define a two-message **P**-certificate system whose first message (still called the CRS for consistency) depends on the statement to be proven.

3.1 P-certificates in the CRS model

We consider the following canonical languages for $\mathbf{P}$: for every constant $c \in N$, let $L_c = \{(M, x, y) : M(x) = y$ within $|x|^c$ steps$\}$. Let $T_M(x)$ denotes the running time of M on input x.

Definition 7 ($\mathbf{P}$-certificate in the CRS model)**.** *A tuple of probabilistic interactive Turing machines,* $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$*, is a* $\mathbf{P}$-certificate system in the CRS model *if there exist polynomials* l_{CRS}*,* l_π*, and the following holds:*

Syntax and Efficiency: *For every* $c \in N$*, every* $q = (M, x, y) \in L_c$*, and every* $k \in N$*, the verification of the statement proceed as follows:*

CRS Generation**:** $\mathsf{CRS} \xleftarrow{\$} \mathsf{Gen}(1^k, c)$*, where* Gen *runs in time* $\text{poly}(k)$*. The length of* CRS *is bounded by* $l_{\mathsf{CRS}}(k)$*.*

Proof Generation**:** $\pi \xleftarrow{\$} \mathsf{P}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q)$*, where* $\mathsf{P}_{\mathsf{cert}}$ *runs in time* $\text{poly}(k, |x|, \min(T_M(x), |x|^c))$ *with* $T_M(x) \leq |x|^c$ *the running time of* M *on input* x*.* The length of the proof π is bounded by $l_\pi(k)$.

Proof Verification**:** $b = \mathsf{V}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q, \pi)$*, where* $\mathsf{V}_{\mathsf{cert}}$ *runs in time* $\text{poly}(k, |q|)$*.*

(Perfect) Completeness: *For every* $c, d \in N$*, there exists a negligible function* μ *such that for every* $k \in N$ *and every* $q = (M, x, y) \in L_c$ *such that* $|q| \leq k^d$*, the probability that in the above execution* $\mathsf{V}_{\mathsf{cert}}$ *outputs 1 is 1.*

Definition 8 (Selective Strong Soundness of $\mathbf{P}$-certificate in CRS model)**.** *We say that a* $\mathbf{P}$*-certificate system* $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$ *is (selectively) strong sound if the following holds:*

- Strong Soundness*: There exists some "nice" super-polynomial function*[6] $T(k) \in k^{\omega(1)}$ *and some "nice" super-constant function*[7] $C(\cdot) \in \omega(1)$ *such that for every probabilistic algorithm* P^* *with running-time bounded by* $T(\cdot)$*, there exists a negligible function* μ*, such that, for every* $k \in N$*,* $c \leq C(k)$*,*

$$\Pr\left[\begin{array}{rcl} (q, \mathbf{st}) & \xleftarrow{\$} & P^*(1^k, c) \\ \mathsf{CRS} & \xleftarrow{\$} & \mathsf{Gen}(1^k, c) \\ \pi & \xleftarrow{\$} & P^*(\mathbf{st}, \mathsf{CRS}) \end{array} : \mathsf{V}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q, \pi) = 1 \wedge q \notin L_c\right] \leq \mu(k)$$

Definition 9 (Uniqueness of $\mathbf{P}$-certificate in the CRS model)**.** *We say that a* $\mathbf{P}$*-certificate system* $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$ *is unique if for every* $k \in N$*, every constant* $c \in N$*, string* $\mathsf{CRS} \in \{0,1\}^*$ *and string* $q \in \{0,1\}^*$*, there exists* at most one *string* $\pi \in \{0,1\}^*$*, such that* $\mathsf{V}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q, \pi) = 1$.

3.2 Two-message P-certificates

The only difference of a two-message $\mathbf{P}$-certificate system from a $\mathbf{P}$-certificate system in the CRS model is that the generation of the CRS (or more precisely the message from the verifier) depends on the statement to be proven. We describe the syntax and efficiency requirement below, with the difference highlighted with underline.

[6]For instance, $T(n) = n^{\log\log\log n}$.
[7]For instance, $C(k) = \log\log\log n$.

Definition 10 (Two-Message **P**-certificate)**.** *A tuple of probabilistic interactive Turing machines,* $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$*, is a* (Two-Message) **P**-certificate system *if there exist polynomials* l_{CRS}*,* l_π*, and the following holds:*

Syntax and Efficiency: *For every* $c \in N$*, every* $q = (M, x, y) \in L_c$*, and every* $k \in N$*, the verification of the statement proceed as follows:*

- CRS Generation: $\mathsf{CRS} \xleftarrow{\$} \mathsf{Gen}(1^k, c, q)$*, where* Gen *runs in time* $\mathrm{poly}(k, |q|)$*. The length of* CRS *is bounded by* $l_{\mathsf{CRS}}(k)$*.*
- Proof Generation: $\pi \xleftarrow{\$} \mathsf{P}_{\mathsf{cert}}(1^k, c, q, \mathsf{CRS})$*, where* $\mathsf{P}_{\mathsf{cert}}$ *runs in time* $\mathrm{poly}(k, |x|, \min(T_M(x), |x|^c))$ *with* $T_M(x) \leq |x|^c$ *the running time of* M *on input* x*.* The length of the proof π is bounded by $l_\pi(k)$.
- Proof Verification: $b = \mathsf{V}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q, \pi)$*, where* $\mathsf{V}_{\mathsf{cert}}$ *runs in time* $\mathrm{poly}(k, |q|)$*.*

(Perfect) Completeness: *The same as in Definition 7.*

The selective strong soundness property of a two-message **P**-certificate system is the same as in Definition 8 except that now the CRS generation algorithm Gen will take the statement q that P^* chooses as an input. Additionally, the uniqueness property is identical to that of Definition 9.

3.3 Delegatable CRS Generation

Definition 11 (Delegatable CRS Generation)**.** *We say that a (two-message)* **P***-certificate* $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$ *has delegatable CRS generation if the CRS generation algorithm* Gen *consists of three subroutines* $(\mathsf{Setup}, \mathsf{PreGen}, \mathsf{CRSGen})$*, and there are polynomials* l_d *and* l_κ*, such that, the following holds:*

Delegatable CRS Generation: $\mathsf{Gen}(1^k, c, q)$ *proceeds in the following three steps:*

1. Generate parameters: $(PP, K) \xleftarrow{\$} \mathsf{Setup}(1^k, c)$*, where* Setup *is probabilistic and runs in time* $\mathrm{poly}(k)$*. We call* PP *the public parameter and* K *the key.*
2. (Public) statement processing: $d = \mathsf{PreGen}(PP, q)$*, where* PreGen *is deterministic and runs in time* $\mathrm{poly}(k, |q|)$*, and the length of* d *is bounded by* $l_d(k)$*. We call* d *the digest of the statement.*
3. (Private) CRS generation: $\kappa \xleftarrow{\$} \mathsf{CRSGen}(PP, K, d)$*, where* CRSGen *is probabilistic and runs in time* $\mathrm{poly}(k)$*, and the length of* κ *is bounded by* $l_\kappa(k)$*.*

Finally, Gen *outputs* $\mathsf{CRS} = (PP, \kappa)$*.*

The reason that we say such a CRS generation procedure is delegatable is because the only part of computation that depends on the statement is the statement processing step; all other steps runs in time a fixed polynomial in the security parameter. However, the statement processing step depends only on the public parameter and the statement; hence to ensure soundness, one only needs to ensure the correctness of this computation, without ensuring the "secrecy" of the computation. Therefore, we also call this step "public" statement processing.

Simple Verification Procedure: Finally, we define an additional property of **P**-certificates: We say that the verification algorithm of a **P**-certificate system is *simple* if $\mathsf{V}_{\mathsf{cert}}$ only depends on the security parameter 1^k, the CRS CRS and the proof π (independent of the statement q and the language index c). Naturally, the uniqueness property of this instantiation is that for any 1^k and CRS string CRS, there is at most one unique accepting proof.

In the next subsection, we show how to instantiate a delegatable **P**-certificate system, using the recent construction of "message hiding encoding" by [KLW14]. Since the instantiation does have a simple verification system, for convenience, in the rest of the paper, we assume by default that a delegatable **P**-certificate system has a simple verification algorithm. We remark that this simplification makes the construction of CZK protocols slightly simpler, but is not necessary.

3.4 Instantiation of P-certificates with Delegatable CRS Generation

Our instantiation relies on the "message hiding encoding" introduced in the recent work by Koppula, Lewko and Waters [KLW14], as a step towards constructing (succinct) indistinguishability obfuscation for Turing machines. Roughly speaking, a message hiding encoding scheme proceeds as follows: Given any message msg (usually generated at random in applications), it transforms a Turing machine computation, M on input x (with time bound T), into an encoding enc, which when decoded yields msg if $M(x) = 1$ (in T steps) and $\perp$ otherwise; on the other hand, the security of the message hiding encoding guarantees that the encoding enc for a non-accepting computation (M, x) hides the message msg. Below, we recall their definition: Let $\Pi^T_M(x)$ denote the Turing machine that runs $M(x)$ for T steps and outputs 1 if the computation accepts and $\perp$ otherwise.

Definition 3 (Message Hiding Encoding [KLW14]). *A message hiding encoding scheme* MHE *consists of two* PPT *algorithms* $(\mathsf{MHE.enc}, \mathsf{MHE.dec})$ *satisfying the following properties*

Syntax and Efficiency: *For any Turing machine M, input* $\mathsf{inp} \in \{0,1\}^*$*, message* $\mathsf{msg} \in \{0,1\}^*$*, time bound $T \in \mathbb{N}$, and security parameter $k \in \mathbb{N}$,*

1. *Encoding: The encoding algorithm* $\mathsf{MHE.enc}(1^k, M, T, \mathsf{inp}, \mathsf{msg})$ *outputs an encoding* enc*, in time* $\mathrm{poly}(k, |M|, |\mathsf{inp}|, |\mathsf{msg}|, \log T)$ *(independent of the running time of the computation.)*
2. *Decoding: The decoding algorithm* $\mathsf{MHE.dec}(1^k, M, \mathsf{inp}, T, \mathsf{enc})$ *outputs a message* msg *or $\perp$, in time* $\mathrm{poly}(k, |M|, |\mathsf{inp}|, \log T, \min(T_M(x), T))$*, where $T_M(x)$ is the running time of M on input x.*

Correctness: *For any Turing machine M, input* $\mathsf{inp} \in \{0,1\}^*$*, message* $\mathsf{msg} \in \{0,1\}^*$*, time bound $T \in \mathbb{N}$, and security parameter $k \in \mathbb{N}$, if $\Pi^T_M(x) = 1$, then*

$$\mathsf{MHE.dec}(1^k, M, \mathsf{inp}, T, \mathsf{MHE.enc}(1^k, M, T, \mathsf{inp}, \mathsf{msg})) = \mathsf{msg}$$

Definition 4 (Message Hiding Property). *A message hiding encoding scheme* MHE *is secure if for every* PPT *adversary A^*, and polynomial Γ, there is a negligible function ε, such that, for every security parameter $k \in \mathbb{N}$, every messages* $\mathsf{msg}_0, \mathsf{msg}_1 \in \{0,1\}^k$*, M of description size at most k, time bound $T \le p(k)$, and input* $\mathsf{inp} \in \{0,1\}^{p(k)}$*, such that,* $\Pi^T_M(\mathsf{inp}) = 0$*, it holds that,*

$$\Pr\left[\begin{array}{lcl} b \xleftarrow{\$} \{0,1\} & & |\mathsf{msg}_0| = |\mathsf{msg}_1| = k \\ (\mathbf{st}, \mathsf{msg}_0, \mathsf{msg}_1, M, T, \mathsf{inp}) \xleftarrow{\$} A^*(1^k) & : & \wedge\ T \le \Gamma(k) \\ \mathsf{enc} \xleftarrow{\$} \mathsf{MHE.enc}(1^k, M, T, \mathsf{inp}, \mathsf{msg}_b) & & \wedge\ A^*(\mathbf{st}, \mathsf{enc}) = b \end{array}\right] \le 1/2 + \varepsilon(k)$$

Furthermore, MHE *is super-polynomially secure if there exists a super-polynomial functions* Γ'*, such that the above condition holds for every* Γ'*-time adversary and function* Γ'*.*

The message hiding encoding is similar to and can be viewed as a weakening of randomized encoding [IK00] in the following sense: The encoding enc for M, x with message msg, can also be viewed as an encoding for the augmented Turing machine $\tilde{M}(x, \mathsf{msg})$ that outputs msg if $M(x) = 1$ and $\perp$ otherwise; while randomized encoding guarantees the privacy of the whole input (x, msg), the message hiding encoding only guarantees privacy of a part of the input msg.

In [KLW14], a construction of a message hiding encoding is proved assuming the existence of indistinguishability obfuscation for circuits and one-way function.

Theorem 1. *Assume the existence of an indistinguishability obfuscation for* $\mathbf{P}/poly$ *and an injective pseudo-random generator (that are super-polynomially secure)*[8]*, there is a message hiding encoding scheme (that is super-polynomially secure).*

P-certificates from Message Hiding Encoding: In is known that randomized encoding (and its slightly enhanced variant of garbling schemes) can be used to ensure the correctness of a computation, as explored in many previous works, for example in [GGP10, BHR12b, BHR12a] for delegation of computation. In fact, for ensuring correctness, it suffices to use a "message hiding encoding" as observed in [KLW14]. Here, the message msg can be viewed as the correctness proof, and the message hiding property ensures that a prover can only obtain msg if the underlying computation is accepting, which implies computational soundness. This naturally suggests a two message proof system for P: Let $\mathsf{Ver}(c, q)$ for $q = (M, x, y)$ be the universal verification algorithm that verifies if $M(x) = y$ in $|x|^c$ steps; it outputs 1 if so and 0 otherwise; it is easy to see that the run time of Ver is bounded by $\alpha|x|^c$ with a universal constant α.

CRS Generation $\mathsf{Gen}(1^k, c, q)$**:** Sample $\pi \xleftarrow{\$} \{0,1\}^k$ at random. Compute the message hiding encoding $\mathsf{enc} \xleftarrow{\$} \mathsf{MHE.enc}(1^k, M = \mathsf{Ver}, T = \alpha|x|^c, \mathsf{inp} = q, \mathsf{msg} = \pi)$, with π as the message. Additionally compute $y = f(\pi)$ using a injective one-way function f. Outputs CRS string $\mathsf{CRS} = (\mathsf{enc}, y)$.

Proof Generation $\mathsf{P_{cert}}(1^k, c, q, \mathsf{CRS})$**:** Parse $\mathsf{CRS} = (\mathsf{enc}, y)$. Decode $z = \mathsf{MHE.dec}(1^k, \mathsf{Ver}, |q|^c, q, \mathsf{enc})$. If $f(z) = y$, output proof $\pi = z$; otherwise, output $\perp$.

Proof Verification $\mathsf{V_{cert}}(1^k, \mathsf{CRS}, \pi)$**:** Parse $\mathsf{CRS} = (\mathsf{enc}, y)$. Accept if $f(\pi) = y$, and reject otherwise.

Efficiency: The proof verification algorithm $\mathsf{V_{cert}}$ runs in strict polynomial time. The complexity of the CRS and proof generation is determined by the complexity of the encoding and decoding algorithm of the message hiding encoding scheme: It follows from the the efficiency of $\mathsf{MHE.enc}$ that Gen runs in time $\mathrm{poly}(k, |\mathsf{Ver}|, |q|, |\pi|, \log(\alpha|x|^c)) = \mathrm{poly}(k, |q|)$, and from the efficiency of $\mathsf{MHE.dec}$ that $\mathsf{P_{cert}}$ runs in time $\mathrm{poly}(k, |\mathsf{Ver}|, |q|, |\pi|, \log(|q|^c), \min(t^*, \alpha|x|^c)) = \mathrm{poly}(k, |q|, \min(t^*, |x|^c))$, where t^* is the running time of M on input x. Moreover, the length of the proof is exactly $|\pi| = k$. In summary, the above system satisfies the efficiency requirement of $\mathbf{P}$-certificates.

Strong Soundness: It follows directly from standard techniques that the message hiding property of MHE implies that for any constant c, the above system is secure against any PPT cheating

[8]The construction of [KLW14] makes use of an IO for $\mathbf{P}/poly$, injective PRG, (selectively secure) puncturable PRF, and an IND-CPA secure public key encryption scheme. All the building blocks exist assuming IO for $\mathbf{P}/poly$ and injective PRG.

prover trying to prove a statically chosen false statement q w.r.t. language L_c. This is because, for a false statement, the computation $\mathsf{Ver}(c, q)$ is not accepting. Thus, it follows form the message hiding property of MHE that, the honest encoding enc of Ver with input (c, q) and message π is indistinguishable from an encoding enc' of $\mathsf{Ver}, (c, q)$ and a different message, say, 0^n. Therefore, if a cheating prover can produce a valid proof for q when receiving an honest $\mathsf{CRS} = (\mathsf{enc}, f(\pi))$ with polynomial probability, it can still produce a valid proof when receiving $\mathsf{CRS}' = (\mathsf{enc}', f(\pi))$. Since a valid proof is π, the cheating prover violates the one-wayness of f. Thus soundness holds.

To obtain strong soundness, we rely on complexity leveling. Assume that MHE and the injective one-way function is super-polynomially secure w.r.t. to a super-polynomial function Γ. There must exist another super-polynomial function Γ' and a super-constant function β', such that, $\Gamma'(k)^{\beta'(k)} \leq \Gamma(k)$ (for example, let Γ' be equal to $2^{\beta(k)\log k}$ for $\beta(k) = \omega(1)$; set $\beta'(k) = \beta(k)^{1/2}$ and $\Gamma'(k) = 2^{\beta'(k)\log k}$). It follows from the same argument that the above argument system is sound against all Γ'-time cheating provers who chooses false statement q w.r.t. any language L_c for $c < \beta'(k)$. This implies that the system is strong sound.

Uniqueness: For any CRS string $\mathsf{CRS}(\mathsf{enc}, y)$, it follows from the injectiveness of the one-way function f, that there is at most one string π, such that, $\mathsf{Ver}(1^k, \mathsf{CRS}, \pi) = 1$, that is, $f(\pi) = y$.

Summarizing, we have,

Theorem 2. *Assume the existence of a message hiding encoding scheme and an injective one-way function (that are both super-polynomially secure), there is a (two-message)* **P***-certificate system with (strong) soundness and uniqueness.*

Delegatable CRS Generation. The message hiding encoding scheme of [KLW14] has certain special structure, such that, the resulting construction of **P**-certificates directly have delegatable CRS generation. The special property is that their encoding algorithm can be divided into three steps matching exactly the three steps in delegatable CRS generation:

(i) First, it generates certain public parameters and a key, depending only on the security parameter k and the time bound T. (Namely, this step runs their Setup-Acc and Steup-Itr algorithms; let PP denote the output of these two algorithms and K is a randomly sampled puncturable PRF key).

(ii) Then, the input of the computation x is processed using the security parameter and public parameters to produce a digest of the input; this step is deterministic. (Namely, this step runs their Write-Store, Prep-Write, and Update algorithms iteratively with the input x and the public parameters PP, to compute a digest w of the input. Note that their input processing step also produces a processed input denoted as *store*, which in an overly simplified view, is similar to a Merkle Hash tree built with leaves x[9]; and *store* is also a part of the encoding. However, we notice that the rest of the encoding does not depend on *store*, and since it can be re-computed by the decoder given x and the public parameter, it can hence be omitted from the encoding.)

(iii) Finally, the encoding is produced depending only on the security parameter, the digest of the input, the public parameter, and the key. (Namely, this step runs the Setup-Spl, Sign-Spl using the PRF key K and the digest w, and then obfuscates using IO a program that depends on the TM M, the time bound T, the public parameter PP and K.)

[9]The actual computation of *store* is much more complicated. In an over-simplified view, it is similar to a Merkle hash tree computed using a specially crafted hash function implemented using IO.

These three steps for generating an encoding corresponds exactly to the Setup, PreGen and CRSGen algorithms in a delegatable CRS generation, with the CRSGen additionally computes the image $y = f(\pi)$. Thus, combining Theorem 2 with the construction of message hiding encoding of [KLW14], and noticing the special structure of its encoding algorithm, we have,

Corollary 1. *Assume the existence of a message hiding encoding scheme and an injective pseudo-random generator (that are both super-polynomially secure), there is a (two-message)* **P***-certificate system with (strong) soundness, uniqueness, and delegatable CRS generation.*

4 Our Protocol

We proceed to describe formally our protocol, (P, V). The protocol relies on the following primitives:

- A non-interactive *perfectly binding* commitment scheme com. We assume without loss of generality that com only needs n bits of randomness to commit to any n-bit string, (as it can always expand these n bits into a longer sequence using a PRG).

  The requirement for a *perfectly binding* commitment scheme can be weakened to rely only on a *statistically binding* commitment scheme. See Remark 2 for more details.

- A strong (two-message) **P**-certificate system $(\mathsf{Gen}, \mathsf{P}_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$ with delegatable CRS generation $\mathsf{Gen} = (\mathsf{Setup}, \mathsf{PreGen}, \mathsf{CRSGen})$ (and simple verification). The strong soundness property is associated with parameter $T(\cdot)$ and $C(\cdot)$, where $T(\cdot)$ is a "nice" super-polynomial function and $C(\cdot)$ is a "nice" super-constant function. The uniqueness property ensures that for every string CRS, there exists at most one proof π that is accepted by $\mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}, \pi) = 1$. This allows us to define the following deterministic oracle $\mathcal{O}^n_{Vcert}$, which will be used in the CZK protocol later.

  $$\mathcal{O}^n_{Vcert}(\mathsf{CRS}) = \begin{cases} \pi & \text{If there exists unique} \pi \text{ s.t. } \mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}, \pi) = 1 \\ \perp & \text{otherwise} \end{cases}$$

  We call $\mathcal{O}^n_{Vcert}$ the **P**-certificate oracle. Additionally, we consider a universal emulator $\mathsf{Emulator}^n$ that on input (P, x, O) emulates the execution of a *deterministic oracle machine* P on input x with oracle $\mathcal{O}^n_{Vcert}$ as follows: It parses O as an vector; to answer the i^{th} query CRS_i from P, it checks whether O_i is the right answer from this CRS (i.e., $\mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}_i, O_i)$ $= 1$); if so, it returns O_i to P; otherwise, it aborts and outputs $\perp$. Finally, the emulator outputs the output of P.

  For simplicity, we assume that the lengths of the CRS, the proof π, and the digest of statement d are all bounded by n, the security parameter. This is without loss of generality, and can be achieved by scaling down the security parameter.

  We assume by default that the two message **P**-certificate system has a simple verification procedure (i.e., $\mathsf{V}_{\mathsf{cert}}$ depends only on $1^k, \mathsf{CRS}, \pi$, but not the statement). This is without loss of generality, since our instantiation based on the message hiding encoding of [KLW14] satisfies this property. But this is not necessary. See remark 3 on how to avoid using this property.

- A family of hash functions $\{\mathcal{H}_n\}_n$: to simplify the exposition, we here assume that both com and $\{\mathcal{H}_n\}_n$ are collision resistant against circuits of size $T'(\cdot)$, where $T'(\cdot)$ is "nice" super-polynomial function.

As in [BG08], this assumption can be weakened to just collision resistance against polynomial-size circuits by modifying the protocol to use a "good" error-correcting code $\mathtt{ECC}$ (i.e., with constant distance and with polynomial-time encoding and decoding), and replace commitments $\mathsf{com}(h(\cdot))$ with $\mathsf{com}(h(\mathtt{ECC}(\cdot)))$. See Remark 1 for more discussion.

- An indistinguishability obfuscator $i\mathcal{O}$ for circuits.
- A constant-round WIUA argument system, a constant-round $\mathcal{WISSP}$ proof system, and a constant-round $\mathcal{ZK}$ argument system.

Let us now turn to specifying the protocol (P, V). The protocol makes use of three parameters: $m(\cdot)$ is a polynomial that upper bounds the number of concurrent sessions; $\Gamma(\cdot)$ is a "nice" super-polynomial function such that $T(n), T'(n) \in \Gamma(n)^{\omega(1)}$, and $D(\cdot)$ is a "nice" super-constant function such that $D(n) \leq C(n)$. Let $m = m(n)$, $\Gamma = \Gamma(n)$ and $D = D(n)$. In the description below, when discussing $\mathbf{P}$-certificates, we always consider the language L_D. For simplicity, below we do not explicitly discuss about the length of the random strings used by various algorithms.

The prover P and the verifier V, on common input 1^n and x and private input a witness w to P, proceed as follow:

Phase 1–Program Slot: P and V exchanges the following three messages.

(a) V chooses a randomly sampled hash function $h \leftarrow \mathcal{H}_n$.

(b) P sends a commitment c to 0^n using com, and random coins ρ_1.

(c) V replies with a random "challenge" r of length $4n$.

We call (c, r) the program-slot.

NOTE: *In the simulation, the simulator commits to a program $\tilde{S}_1$.*

Phase 2—Commit to Statement: P and V exchanges the following messages.

(a) P sends a commitment c_2 to 0^n using com, and random coins ρ_2.

(b) P gives a WIUA argument of the statement that either $x \in L$ OR there exists $\tilde{S}_1 \in \{0,1\}^{\Gamma(n)}$, $j \in [m]$, $s \in \{0,1\}^n$, $\pi \in \{0,1\}^n$, $\sigma \in \{0,1\}^{\Gamma(n)}$, ρ, ρ_2 such that,

Knowledge of Statement: $c_2 = \mathsf{com}(h(q); \rho_2)$, where $q \in \{0,1\}^{3\Gamma}$.

Correctness of Statement: The statement q satisfy the following properties:

- USE OF EMULATOR: q can be parsed into $(\mathsf{Emulator}^n, (\tilde{S}_1, (1^n, j, s), \sigma), r)$.
- PROGRAM CONSISTENCY: $c = \mathsf{com}(h(\tilde{S}_1); \rho)$.

If the argument is not accepting, V aborts.

NOTE: *By definition of the emulator $\mathsf{Emulator}^n$, on input $(\tilde{S}_1, (1^n, j, s), \sigma)$, it will emulate the execution of the deterministic oracle machine $\tilde{S}_1(1^n, j, s)$ with oracle $\mathcal{O}^n_{V cert}$ using answers stored in vector σ.*

The purpose of this phase is twofold: First, it enforces a cheating prover to commit to the "trapdoor" statement before the CRS of the $\mathbf{P}$-certificate is generated, and hence the soundness of the protocol only relies on the selective *soundness of the $\mathbf{P}$-certificate. Second, it checks whether the "trapdoor" statement has the right structure, in particular, the statement is about whether $\tilde{S}_1^{\mathcal{O}_{V cert}}(1^n, j, s) = r$, when the oracle is emulated by $\mathsf{Emulator}^n$ using σ, who checks the correctness of the proofs in σ.*

Note that the soundness of the protocol will crucially rely on the fact that the input to $\tilde{S}_1$ has length at most $3n$, much smaller than the length, $4n$, of the output r (and the deterministic oracle $\mathcal{O}_{Vcert}$ is emulated correctly by $\mathsf{Emulator}^n$*). On the other hand, in the simulation, the simulator will commit to the "trapdoor" statement,* $q = (\mathsf{Emulator}^n, (\tilde{S}_1, (1^n, j, s), \sigma), r)$ *in order to "cheat".*

Phase 3—Delegate Public Statement Processing: V delegates the public statement processing to P:

(a) V generates $(PP, K) = \mathsf{Setup}(1^n, D; \rho_{\mathsf{Setup}})$ using random coins r_{CRS}, and sends PP.

(b) P sends a commitment c_3 to 0^n using com, and random coins ρ_3.

(c) P gives a WIUA argument of the statement that either $x \in L$ OR there exists, $d \in \{0,1\}^n$, $q \in \{0,1\}^{3\Gamma}$, ρ_2, ρ_3, such that,

Statement Consistency: $c_2 = \mathsf{com}(h(q); \rho_2)$.

Digest Consistency: $c_3 = \mathsf{com}(d; \rho_3)$.

Correctness of Digest: $d = \mathsf{PreGen}(PP, q)$.

If the argument is not accepting, V aborts.

NOTE: *The purpose of this Phase is to allow the verifier to delegate the computation of the digest of the statement to P. In simulation, the simulator will compute, commit to and prove correctness of* $d = \mathsf{PreGen}(PP, q)$*. V cannot compute d itself, since (1) it does not know the "trapdoor" statement q and (2) the computation takes* $\mathrm{poly}(n, |q|)$*, which is too expensive for the verifier.*

Phase 4—Delegate Private CRS Generation: V delegates the private CRS generation to P:

(a) V sends the indistinguishability obfuscation $\Lambda \xleftarrow{\$} i\mathcal{O}(\mathbf{P})$ of program $\mathbf{P} = \mathbf{P}^{n, c_3, PP, K, \rho_{\mathsf{CRSGen}}}$ with c_4, K, and a random string ρ_{CRSGen} hardwired in. $\mathbf{P}$ on input (d', ρ') checks whether $c_3 = \mathsf{com}(d', \rho')$ and outputs $\kappa = \mathsf{CRSGen}(PP, K, d; \rho_{\mathsf{CRSGen}})$ if it is the case, and $\perp$ otherwise. The functionality of $\mathbf{P}$ is described formally in Figure 2.

Circuit $\mathbf{P} = \mathbf{P}^{n, c_3, PP, K, \rho_{\mathsf{CRSGen}}}$**:** On input (d', ρ') where $d' \in \{0,1\}^n$ and $\rho' \in \{0,1\}^n$, does:

(a) Check if $c_3 = \mathsf{com}(d'; \rho')$; if not, output $\perp$.

(b) Otherwise output $\kappa = \mathsf{CRSGen}(PP, K, d'; \rho_{\mathsf{CRSGen}})$.

Circuit $\mathbf{Q} = \mathbf{Q}^{n, c_3, \kappa}$**:** On input (d', ρ') where $d' \in \{0,1\}^n$ and $\rho' \in \{0,1\}^n$, does:

(a) Check if $c_3 = \mathsf{com}(d'; \rho')$; if not, output $\perp$.

(b) Otherwise output κ.

The above circuits are padded to their maximum size.

Figure 2: Circuits used in the construction and proof of CZK protocol $\langle P, V \rangle$

(b) V gives a $\mathcal{ZK}$ argument of the statement that there exists $K \in \{0,1\}^n$, ρ_{Setup}, ρ_{CRSGen}, $\rho_{i\mathcal{O}}$, such that,

Correctness of Public Parameter: $(PP, K) = \mathsf{Setup}(1^n, D; \rho_{\mathsf{Setup}})$.
Correctness of Obfuscation: $\Lambda = i\mathcal{O}(\mathbf{P}^{c_3, PP, K, \rho_{\mathsf{CRSGen}}}; \rho_{i\mathcal{O}})$

If the argument is not accepting, P aborts.

(c) P sends commitment c_4 of 0^n using com and random coins ρ_4.

(d) P gives a $\mathcal{WISSP}$ proof of the statement that either $x \in L$ OR there exists $\mathsf{CRS} \in \{0,1\}^n$, $d' \in \{0,1\}^n$, ρ', ρ_4, such that,

CRS Consistency: $c_4 = \mathsf{com}(\mathsf{CRS}; \rho_4)$.
Correctness of CRS: $\mathsf{CRS} = (PP, \kappa)$ and $\kappa = \overline{\mathbf{P}}(d', \rho')$.

If the proof is not accepting, V aborts.

Note: *The purpose of this Phase is to allow the verifier to delegate the computation of* CRS *to* P*. In simulation, the simulator will compute, commit to, and prove correctness of* $\mathsf{CRS} = (PP, \kappa)$*, with* $\kappa = \overline{\mathbf{P}}(d, \rho_3)$*. V cannot compute κ itself, even though the computation takes only polynomial time in n, since d cannot be revealed to V in order to ensure the indistinguishability of the simulation. On the other hand, to ensure the "privacy" of the CRS computation, V delegates this computation via obfuscation.*

Phase 5—Final Proof: P gives the final proof:

(a) P gives a $\mathcal{WISSP}$ proof of the statement that either $x \in L$ OR there exists $\pi \in \{0,1\}^n$, $\mathsf{CRS} \in \{0,1\}^n$, ρ_4, such that,

CRS Consistency: $c_4 = \mathsf{com}(\mathsf{CRS}; \rho_4)$,
Proof Verification: π verifies w.r.t. CRS, $\mathsf{V_{cert}}(1^n, \mathsf{CRS}, \pi) = 1$.

V accepts if the proof is accepting.

Note: *In the simulation, the simulator will compute the proof* $\pi \xleftarrow{\$} \mathsf{P_{cert}}(1^k, D, q, \mathsf{CRS})$*, and succeed in the final proof by using π and* CRS, ρ_4 *generated in the last phase as "trapdoor" witness.*

Theorem 3. *Assume indistinguishability obfuscation for* $\mathbf{P}$*/poly, an injective pseudo-random generator, and collision resistant hash functions that are super-polynomially secure. Then, the above protocol $\langle P, V \rangle$ is a concurrent $\mathcal{ZK}$ argument system for* NP*.*

The completeness of the protocol follows from the completeness of the WIUA argument of knowledge, $\mathcal{WISSP}$, and the $\mathcal{ZK}$ argument. Below, we prove first the concurrent zero knowledge property and then the soundness of the protocol.

4.1 Proof of Concurrent Zero-Knowledge

The goal of our simulator is to try to "commit to its own code" and prove about its own execution using $\mathbf{P}$-certificates in a way that prevents a blow-up in the running-time. Note that the only expensive part of this process is the generation of the $\mathbf{P}$-certificates $\vec{\pi}$; the rest of the computation has *a-priori* bounded complexity (depending only on the size and running-time of V^*). To take advantage of this observation, we thus have the simulator only commit to an oracle program that generates prover messages (in identically the same way as the actual simulator), but getting certificates $\vec{\pi}$ from the $\mathbf{P}$-certificate oracle $\mathcal{O}_{V cert}$.

In more detail, to describe the actual simulator S, let us first describe two "helper" simulators S_1, S_2. Roughly speaking, S_1 is an interactive machine that simulates prover messages in a "right" interaction with V^*. Additionally, S_1 excepts to have access to oracle $\mathcal{O}_{Vcert}$ on the "left", in particular, at any point, it can send a CRS string CRS and gets back the $\pi = \mathcal{O}_{Vcert}(\mathsf{CRS})$ the unique accepting certificate w.r.t. this CRS (or $\perp$, if such a certificate does not exist); the oracle will be simulated by S_2, who provides these "left" certificates.

Let us turn to a formal description of the S_1 and S_2. To simplify the exposition, we assume w.l.o.g that V^* has its non-uniform advice z hard-coded, and is deterministic (as it can always get its random tape as non-uniform advice).

On a high-level, $S_1(1^n, x, M, s, \ell)$ acts as a prover in a "right" interaction, communicating with a concurrent verifier V^*, while accessing oracle on the "left". (The input x is the statement to be proved, the input M will later be instantiated with the code of S_1, and the input (s, ℓ) is used to generate the randomness for S_1; s is the seed for the forward secure pseudorandom generator g, and ℓ is the number of n-bit long blocks to be generated using g.) A communication round in the "right" interaction with V^* refers to a verifier message (sent by V^*) followed by a prover message (sent by S_1).

PROCEDURE OF SIMULATOR S_1: Let us now specify how S_1 generates prover messages in its "right" interaction with V^*. $S_1^{\mathcal{O}_{Vcert}}(1^n, x, M, s, \ell)$ acts as follows:

Generate Randomness: Upon invocation, S_1 generates its "random-tape" by expanding the seed s; more specifically, let $(s_\ell, s_{\ell-1}, \ldots s_1)$, $(q_\ell, q_{\ell-1}, \ldots, q_1)$ be the output of $g(s, \ell)$. We assume without loss of generality that S_1 only needs n bits of randomness to generate any prover message (it can always expand these n bits into a longer sequence using a PRG); in order to generate its j^{th} prover message, it uses q_j as randomness.

Simulate Phase 1—"Commit to its own code": Upon receiving a hash function h_i in session i during the j^{th} communication round, S_1 provides a commitment c_i to (the hash of) the deterministic oracle machine $\tilde{S}_1(1^n, \alpha, s') = wrap(M(1^n, x, M, s', \alpha), V^*, \alpha)$, where $wrap(A, B, \alpha)$ is the program that lets A communicate with B for α rounds, while allowing A to access oracle $\mathcal{O}_{Vcert}$, and finally outputting B's message in the j^{th} communication round.

NOTE: *That is, $\tilde{S}_1(1^n, \alpha, s', \tau)$ emulates α rounds of an execution between S_1 and V^* where S_1 expands out the seed s' into α blocks of randomness and additionally have access to $\mathcal{O}_{Vcert}$.*

Simulate Phase 2—"Commit to the trapdoor statement": Upon receiving a challenge r_i in session i during the j^{th} communication round, S_1 needs to commit to the "trapdoor" statement it will later prove in the final proof. To do so, it prepares statement $q_i = (\mathsf{Emulator}^n, (\tilde{S}_1, (1^n, j, s_j), \tau_{j-1}), r_i)$, where τ_{j-1} is the list of oracle answers received by S_1 in the first $j-1$ communication rounds.

NOTE: *That is, the "trapdoor" statement is that the execution of $\tilde{S}_1(1^n, j, s_j)$, emulated by $\mathsf{Emulator}^n$, outputs r, when its k^{th} oracle queries is answered using $\tau_{j-1,k}$; additionally, the validity of each answer is checked by $\mathsf{Emulator}^n$ (i.e., the answer must be an accepting proof w.r.t. the query CRS string).*

By construction of $\tilde{S}_1$, this means after j communication rounds between S_1 and V^*, where S_1 uses randomness expanded out from s_j, and oracle answers τ_{j-1}, V^* outputs r_i in the j^{th}s communication round. Note that since we only require $\tilde{S}_1$ to generate the j^{th} verifier message, giving him the seed (s_j, j) as input suffices to generate all prover messages in rounds $j' < j$. It follows from the consistency requirement of the forward secure PRG that $\tilde{S}_1$ using (s_j, j)

as seed will generate the exact same random sequence for the $j-1$ first blocks as if running $\tilde{S}_1$ using (s,ℓ) as seed. Therefore, the "trapdoor" statement holds.

In later communication rounds, when S_1 receives a message from V^* belonging to the WIUA in Phase 2 of session i, S_1 proves honestly that it knows the statement q_i it is committing to in session i, and the statement is correctly formatted and consistent with the program $\tilde{S}_1$ committed to in Phase 1 of session i.

Simulate Phase 3— "Process the trapdoor statement": Upon receiving a public parameter PP_i in session i during the j^{th} communication round, S_1 needs to commit to the digest d_i of the "trapdoor" statement q_i of session i. To do so, it computes honestly $d_i \xleftarrow{\$} \mathsf{PreGen}(PP_i, q_i)$ and commits to d_i using com, and randomness ρ_i.

In later communication rounds, when S_1 receives a message from V^* belonging to the WIUA in Phase 3 of session i, S_1 proves honestly that it knows d_i committed to in Phase 3 of session i and it is computed correctly w.r.t. PP_i and a statement q_i committed to in Phase 2 of session i.

Simulate Phase 4— "Compute the CRS": Upon receiving an obfuscated program Λ_i, S_1 acts as an honest verifier of the $\mathcal{ZK}$ argument to verify that PP_i and Λ_i in session i are correctly generated. Upon receiving the last message of the $\mathcal{ZK}$ argument, in the j^{th} communication round, S_1 needs to commit to the CRS_i of session i. To do so, it computes $\kappa_i = \Lambda_i(d_i, \rho_i)$. If the output is $\perp$, S_1 aborts. Otherwise, it commits to $\mathsf{CRS}_i = (PP_i, \kappa_i)$ using com.

In later communication rounds, when S_1 receives a message from V^* belonging to the $\mathcal{WISSP}$ in Phase 4 of session i, S_1 proves honestly that it knows κ_i committed to in Phase 4 of session i and it is computed correctly w.r.t. Λ_i and a digest d_i committed to in Phase 3 of session i.

Simulate Phase 5— "Prove the trapdoor statement using P-certificate": Upon receiving the last message from V^* in Phase 4 of session i, during the j^{th} communication round, S_1 needs to prove in the $\mathcal{WISSP}$ proof that there is a $\mathbf{P}$-certificate that verifies the validity of the "trapdoor" statement q_i w.r.t. the CRS string CRS_i committed to in Phase 4 of session i. To do so, it sends query CRS_i to its oracle $\mathcal{O}_{Vcert}$, and obtains answer π_i. It aborts if $\pi_i = \perp$. Otherwise, S_1 provides an honest $\mathcal{WISSP}$ that $\mathsf{V_{cert}}(1^n, \mathsf{CRS}_i, \pi_i) = 1$ w.r.t. CRS_i which is the committed value in Phase 4 of session i.

<u>PROCEDURE OF SIMULATOR S_2:</u> $S_2(1^n, x, M, s, \ell)$ internally emulates ℓ messages of an execution between $S_1(1^n, x, M, s, \ell)$ and V^*, and simulates the oracle $\mathcal{O}_{Vcert}$ for S_1. In a communication round j when S_1 sends an oracle query CRS_i for a session i, S_2 generates a certificate π_i of the statement $q_i = (\mathsf{Emulator}^n, (\tilde{S}_1, (1^n, j', s_{j'}), \tau_{j'-1}), r_{j'})$ w.r.t. CRS_i, that is, $\pi_i \xleftarrow{\$} \mathsf{P_{cert}}(1^n, D, q_i, \mathsf{CRS}_i)$ (where j' is the round in which the challenge r_i is sent by V^*, q_i and CRS_i are generated by S_1 (emulated internally by S_2) in Phase 2 and 4 of session i). S_2 checks if indeed $\mathsf{V_{cert}}(1^n, \mathsf{CRS}_i, \pi_i) = 1$, it outputs fail if this is not the case, and otherwise, feeds π_i to S_1. Finally, S_2 outputs its view (which in particular, contains the view of V^*) at the end of the execution.

<u>PROCEDURE OF THE FINAL SIMULATOR S:</u> The final simulator $S(1^n, x)$ simply runs $S_2(1^n, x, S_1, s, T(n+|x|))$, where s is a uniformly random string of length n and $T(n+|x|)$ is a polynomial upper-bound on the number of messages sent by V^* given the common input $1^n, x$, and extracts out and outputs, the view of V^* from the output of S_2. (In case that S_2 outputs fail, S outputs fail as well.)

Running-time of S. Let us first argue that S_1 runs in polynomial time.

1. In Phase 1, it only takes S_1 polynomial-time to generate the commitments (since V^* has a polynomial-length description, and thus also the code of $\tilde{S}_1$).

2. In Phase 2, it also only takes S_1 polynomial time to commit to the statements q_i (since $\mathsf{Emulator}^n$, $(1^n, j, s_j)$, and r have fixed polynomial lengths, and $\tilde{S}_1$ and τ_{j-1} have polynomial length description, depending on the size of V^*). Furthermore, the witnesses of the WIUA in Phase 2 has polynomial length; by the relative prover efficiency condition of the WIUA, each such proof only requires some fixed polynomial-time.

3. In Phase 3, processing the statements q_i takes time polynomial in the length of the statement and n, which is polynomial. Furthermore, committing to the outputs d_i and proving about their correctness using WIUA also takes only polynomial time (by the relative prover efficiency of WIUA).

4. In Phase 4, since the CRS generation is very efficient, taking time polynomial in only the security parameter, S_1 completes all Phase 4 in polynomial time.

5. In Phase 5, the simulator proves about the verification of a $\mathbf{P}$-certificate w.r.t. to a CRS string committed to in Phase 4. Since both steps takes time poly(n), S_1 completes all Phase 5 in polynomial time.

Overall, the whole execution of S_1 takes some fixed polynomial time (in the length of V^* and thus also in the length of x.) It directly follows that also $\tilde{S}_1$'s running-time is polynomially bounded.

Finally, since S_2 is simply providing certificates about the execution of $\tilde{S}_1$, it follows by the relative prover efficiency condition of $\mathbf{P}$-certificates, that S_2 runs in polynomial time, and thus also S.

Indistinguishability of the simulation Fix any cheating verifier V^*, we first argue that during the execution of S for simulating the view of V^*, the probability that S_2 (and hence S) outputs fail is negligible. By construction, S_2 outputs fail when for some session i, the proof π_i that it constructs honestly using $\mathsf{P}_{\mathsf{cert}}$ does not verify w.r.t. the CRS_i that S_1 computes. It follows from the soundness of the $\mathcal{ZK}$ argument in Phase 4 of session i that, with overwhelming probability, V^* in session i computes $(PP, K) \xleftarrow{\$} \mathsf{Setup}(1^n, D)$ and the obfuscation $\Lambda \xleftarrow{\$} i\mathcal{O}(\mathbf{P})$ of $\mathbf{P} = \mathbf{P}^{n,c_3,PP,K,\rho_{\mathsf{CRSGen}}}$ correctly w.r.t. some random strings ρ_{Setup} and $\rho_{i\mathcal{O}}$. In this case, since S_1 evaluates PreGen, commits to the produced digest d_i, and evaluates Λ_i honestly, it follows from the perfect correctness of the indistinguishability obfuscator, the perfect completeness of the $\mathbf{P}$-certificate system, and the perfect binding property of com that as long as q_i is a true statement, S_2 would generate an accepting proof for it w.r.t. CRS_i. By construction, q_i is a true statement. Therefore, the probability that S_2 outputs fail is negligible.

Below we argue about the indistinguishability of the simulation conditioning on that S_2 does not output fail. Assume that there exists a cheating verifier V^*, a distinguisher D and a polynomial p such that the real view and the simulated view of V^* can be distinguished by D with probability $\frac{1}{p(n)}$ for infinitely many n. More formally, for infinitely many $n \in N$, $x \in L \cap \{0,1\}^{\mathrm{poly}(n)}$, $w \in \mathbf{R}_L(x)$ and $z \in \{0,1\}^{\mathrm{poly}(n)}$, it holds that

$$\left|\Pr[D(\mathsf{View}_{V^*} \langle P(w), V^*(z)\rangle (1^n, x)) = 1] - \Pr[D(S(1^n, x, z)) = 1]\right| \geq \frac{1}{p(n)} \tag{1}$$

Consider such n, x, z (and assume that z is hard-coded into the description of V^*), and consider $T = T(n+|x|)$ hybrid experiments (recall that $T(n+|x|)$ is the maximum number of communication rounds given common input $1^n, x$).

- In hybrid H_j, the first j communication rounds are simulated exactly as by S (using pseudo-randomness), but all later communication round $j' > j$ are generated honestly using *true* randomness q'_j being uniformly distributed in $\{0,1\}^n$. More precisely, every prover commitment sent after round j is a commitment to 0^n (i.e., Step 1-(b), 2-(a), 3-(b) and 4-(c)); every WIUA argument (i.e., Step 2-(b), 3-(c)) and every $\mathcal{WISSP}$ proof (i.e., Step 4-(d), 5-(a)) that *start after* round j uses the true witness w instead of "fake" witnesses that S uses; however, every WIUA argument and $\mathcal{WISSP}$ proof that start at or before round j are still proven using (appropriate) "fake" witnesses as S does; importantly, all prover messages generated after round j uses truly random coins.

It follows by Equation 1 and a hybrid argument that there exist some j and a polynomial p'' such that D distinguishes H_j and H_{j+1} with probability $\frac{1}{p''(n)}$. Now, consider another hybrid experiment $\tilde{H}_j$ that proceeds just at H_{j+1}, but where true randomness is used in communication round $j+1$ (but still committing to the the same values as S does and using "fake" witness as S does). It follows by the forward security of the PRG g that the outputs of H_{j+1} and $\tilde{H}_j$ are indistinguishable—the reason we need *forward security* is that to emulate communication rounds $j' \leq j$, the seeds $s_{j'}$ may need to be known (as they are part of the "trapdoor" statements). Indistinguishability of $\tilde{H}_j$ and H_j follows directly by either the hiding property of the commitment scheme (if in the $j+1$ round, the prover message is a commitment), or the witness indistinguishability property of the WIUA or $\mathcal{WISSP}$ (if in this round, the prover message is a message of WIUA or $\mathcal{WISSP}$). It thus leads to a contradiction and completes the proof of the indistinguishability of the simulation.

4.2 Proof of Soundness

We now prove soundness of our protocol. Assume for contradiction that there is a non-uniform deterministic polynomial time cheating prover P^* and a polynomial $p(n)$, such that for infinitely many $n \in \mathbb{N}$, there exists $x \notin L$ such that $\Pr[(P^*, V)(1^n, x) = 1] \geq 1/p(n)$. Let E be the "global" proof-of-knowledge extractor of the WIUA, and E' be the knowledge extractor of the $\mathcal{WISSP}$.

Fix such n and $x \notin L$. Let us consider the following experiment Exp:

- Run $(P^*, V)(1^n, x)$ up to the point where P^* sends c_2. Let $P^*_{\mathsf{prefix}_1}$ be the residual WIUA prover for the first WIUA of the protocol (in Phase 2), resulting from feeding it the messages $\mathsf{prefix}_1 = (h, r)$. Run $w_1 \leftarrow E_{s_1}^{P^*_{h,r}}$, where s_1 is uniform randomness to E. If the extraction fails to extract a valid witness, then output $\perp$.

- Continue to run $(P^*, V)(1^n, x)$ up to the point where P^* sends c_3. Let $P^*_{\mathsf{prefix}_2}$ be the residual WIUA prover for the second WIUA of the protocol (in Phase 3), resulting from receiving verifier's messages prefix_2 (including h, r, WIUA messages, and PP). Run $w_2 \leftarrow E_{s_2}^{P^*_{\mathsf{prefix}_2}}$, where s_2 is uniform randomness to E. If the extraction fails to extract a valid witness, then output $\perp$.

- Continue to run $(P^*, V)(1^n, x)$ up to the point where P^* sends c_4. Let $P^*_{\mathsf{prefix}_3}$ be the residual $\mathcal{WISSP}$ prover for the first $\mathcal{WISSP}$ of the protocol (in Phase 4), resulting from receiving verifier's messages prefix_3. Run $w_3 \leftarrow E'^{P^*_{\mathsf{prefix}_3}}_{s_3}$, where s_3 is uniform randomness to E'. If the extraction fails to extract a valid witness, then output $\perp$.

- Continue to run $(P^*, V)(1^n, x)$ up to the beginning of the final $\mathcal{WISSP}$. Let $P^*_{\mathsf{prefix}_4}$ be the residual $\mathcal{WISSP}$ prover for the second $\mathcal{WISSP}$ of the protocol (in Phase 5), resulting from receiving verifier's messages prefix_4. Run $w_34 \leftarrow E_{s_4}^{\prime P^*_{\mathsf{prefix}_4}}$, where s_4 is uniform randomness to E'. If the extraction fails to extract a valid witness, then output $\perp$.

- Continue to finish $(P^*, V)(1^n, x)$. If the verifier rejects, then output $\perp$. Otherwise, output $\tilde{S}_1$ and q in the witness w_1 of the first WIUA, and the verifier's challenge message r. (Note that since $x \notin L$, so the witness w_1 must be of the form $(q, \tilde{S}_1, j, s, \sigma, \rho, \rho_2)$.)

Let $\mathsf{Not\text{-}bot}$ denote the event that Exp does not output $\perp$. We first argue that $\mathsf{Not\text{-}bot}$ happens with non-negligible probability. Recall that the knowledge extractors of both WIUA and $\mathcal{WISSP}$ guarantee that if the cheating prover convinces the verifier with a non-negligible probability, then the extractor also successes with a non-negligible probability. Since P^* convinces V with probability at least $1/p(n)$, it holds that with probability at least $1/2p(n)$ over $(P^*, V)(1^n, x)$, all residual provers $P^*_{\mathsf{prefix}_i}$ for $i \in [4]$ have success probability at least $1/8p(n)$, and in such case, each extractor succeeds with non-negligible probability. Since the extractors use independent randomness, the probability that V accepts and all extractors succeed (in which case Exp does not output $\perp$) is non-negligible. Let $1/p'(n)$ denote this non-negligible function.

Let $\mathsf{Inconsistent}$ denote the event that the extracted witnesses in Exp are inconsistent. Specifically, this means at least one of the following happens: (i) q in w_1 and w_2 are not equal, (ii) d in w_2 and w_3 are not equal, and (iii) CRS in w_3 and w_4 are not equal. It is not hard to see that $\Pr[\mathsf{Inconsistent}] \leq \mathrm{negl}(n)$, since otherwise, we can break either the binding property of com, or the collision resistant property of CRHs. Let $\mathsf{Consistent}$ be $\neg\mathsf{Inconsistent}$. By a union bound, we have $\Pr[\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent}] \geq 1/p'(n) - \mathrm{negl}(n)$.

Let us now switch to a hybrid experiment Exp', which is identical to Exp, except that (i) the obfuscated program $\Lambda \stackrel{\$}{\leftarrow} i\mathcal{O}(\mathbf{P})$ in the first verifier message in Phase 4 is replaced by $\Lambda \stackrel{\$}{\leftarrow} i\mathcal{O}(\mathbf{Q})$, where the κ hard-wired in $\mathbf{Q}$ is set to be $\kappa = \mathsf{CRSGen}(PP, K, \mathsf{PreGen}(PP, q); \rho_{\mathsf{CRSGen}})$, where q is the statement extracted in both witnesses w_1 and w_2, and (ii) the ZK proof right after is replaced by that generated by the ZK simulator. Note that by soundness of WIUA, c_3 is a commitment of $d = \mathsf{PreGen}(PP, q)$ except with negligible probability. This together with the perfect binding property of com implies that $\mathbf{P}$ and $\mathbf{Q}$ are functionally equivalent (except with negligible probability). Thus, by the security of $i\mathcal{O}$ and zero knowledge property of the ZK argument, Exp and Exp' are indistinguishable. Hence, we have $\Pr[\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent}] \geq 1/p'(n) - \mathrm{negl}(n)$ holds in Exp' as well.

Let $\mathsf{False}\text{-}q$ denote the event that the statement q extracted in w_1 is false. We claim that $\Pr[\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent} \wedge \mathsf{False}\text{-}q] \leq \mathrm{negl}(n)$. Indeed, if $\Pr[\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent} \wedge \mathsf{False}\text{-}q] \geq 1/p''(n)$ for some non-negligible $1/p''(n)$, then we can break soundness of the two-message $\mathbf{P}$-certificate (with delegatable CRS generation) by the following reduction:

- $\mathsf{P}^*_{\mathsf{cert}}$ emulates Exp' internally up to the point of extracting w_1, and outputs q in w_1 as the chosen statement.

- Upon receiving $\mathsf{CRS} = (PP, \kappa)$, where $(PP, K) = \mathsf{Setup}(1^n; \rho_{\mathsf{Setup}})$, $d = \mathsf{PreGen}(PP, q)$, and $\kappa = \mathsf{CRSGen}(PP, K, d; \rho_{\mathsf{CRSGen}})$, $\mathsf{P}^*_{\mathsf{cert}}$ continues to emulate Exp', with PP and κ from CRS, and outputs the extracted π (in Phase 5).

Note that when $(\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent})$ holds, $(\mathsf{P}^*_{\mathsf{cert}}, \mathsf{V}_{\mathsf{cert}})$ emulates Exp' perfectly, and when $(\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent} \wedge \mathsf{False}\text{-}q)$ happens, $\mathsf{P}^*_{\mathsf{cert}}$ convinces $\mathsf{V}_{\mathsf{cert}}$ a false statement. Thus, if $\Pr[\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent} \wedge \mathsf{False}\text{-}q] \geq 1/p''(n)$, then $\mathsf{P}^*_{\mathsf{cert}}$ breaks soundness of $\mathbf{P}$-certificate, a contradiction.

Let $\mathsf{True}\text{-}q = \neg\mathsf{False}\text{-}q$, and $\mathsf{Good} = (\mathsf{Not\text{-}bot} \wedge \mathsf{Consistent} \wedge \mathsf{True}\text{-}q)$. The above claims imply

$$\Pr[\mathsf{Good}] \geq 1/p'(n) - \mathrm{negl}(n).$$

We now derive a contradiction from it. By an averaging argument, this implies that with probability at least $1/3p'(n)$ over the hash function h (the verifier's first message),

$$\Pr[\mathsf{Good}|h] \geq 1/3p'(n).$$

Now, let us consider an experiment Exp_2 that first samples a hash function $h \leftarrow \mathcal{H}$, and then runs Exp' twice with this h and independent fresh randomness. It follows that with probability at least $1/27p'^3(n)$ over Exp_2, the event Good occurs in both executions of Exp'; let us focus on this case. Let $(\tilde{S}_1, q, r)$ and $(\tilde{S}'_1, q', r')$ denote the output of the two branches. By the perfect binding property of com and the collision resistant property of CRHs, we have $\tilde{S}_1 = \tilde{S}'_1$, except with negligible property. Also note that $\mathsf{True}\text{-}q$ implies that there exists two short inputs (j, s) and (j', s') such that $\tilde{S}_1^{\mathcal{O}_{V cert}}(1^n, j, s) = r$ and $\tilde{S}_1^{\mathcal{O}_{V cert}}(1^n, j', s') = r'$. In other words, $\tilde{S}_1^{\mathcal{O}_{V cert}}$ can predict two independent r, r' with probability at least $1/27p'^3(n) - \mathrm{negl}(n)$. However, this is information theoretically impossible, since the fact that $\tilde{S}_1^{\mathcal{O}_{V cert}}$ is deterministic, $|(j, s)| < 3n$, and $|r| = 4n$ implies that $\tilde{S}_1^{\mathcal{O}_{V cert}}$ can only predicts two independent r, r' with exponentially small probability. Hence, we reach a contradiction and complete the soundness proof.

Remark 1. *We remark that the protocol and its soundness proof described above relies on collision resistant hash functions against slightly super-polynomial-sized circuits. This requirement can be weakened to rely on collision resistance against polynomially sized circuits. To do so one should use a "good" error-correcting code* $\mathtt{ECC}$ *(i.e., with constant distance and with polynomial-time encoding and decoding), and replace every commitment of the form* $\mathsf{com}(h(X))$ *with* $\mathsf{com}(h(\mathtt{ECC}(X)))$ *[BG08]. The soundness proof will need to be modified accordingly (while the proof of zero-knowledge remains essentially the same).*

We now briefly sketch the idea. It follows from the global proof of knowledge property of WIUA*, that the witness of each* WIUA *is well-defined, therefore, we can refer to the machine* M *committed in Phase 1, the statement* q *committed in Phase 2, and the statement* q' *w.r.t. which a digest* d *is computed in Phase 3. We first argue that* q, q', M *are all consistent, that is* $q = q'$ *and* q *contains* M*. Suppose not, their encoding through* $\mathtt{ECC}$ *would differ at at least a constant fraction of the coordinates. Then a collision would have been found by relying only on the weak proof of knowledge property of* WIUA *as done in [BG08]. Given that* $q = q'$ *and* M *are consistent, it follows from the same proof as above that by the strong soundness of* $\mathbf{P}$*-certificate (and soundness of* $\mathcal{WISSP}$*), a cheating prover must prove a true statement* q*. Finally, using the same argument as above again, when running the cheating prover twice with the same hash function* h*, with noticeable probability, we will have two executions with two true statements* q_1*,* q_2 *but containing different machines* M_1*,* M_2*; in this case, by relying on* $\mathtt{ECC}$ *and the weak proof-of-knowledge property of* WIUA*, a collision can be found, as done in [BG08]. Therefore, soundness holds assuming only CRHs against polynomial-sized circuits.*

Remark 2. *In the above protocol and analysis, we used a perfectly binding commitment scheme. With some small modification, it suffices to use a 2-message statistically-binding commitment scheme; such a protocol can be obtained from one-way functions [Nao91, HILL99]. To replace a perfectly binding commitment, we modify our protocol to let the verifier sends the receiver's first message* r *of the commitment scheme at the beginning of the protocol. After that, whenever the prover needs to send a commitment to the verifier, it sends the second committer's message w.r.t.*

r. A two-message statistically binding commitment scheme has the property that over the random choice of the receiver's first message, with overwhelming probability, the committer's message determines the committed value. Therefore, with overwhelming probability over the choice of r, the second committer's message can be used as a perfectly binding commitment scheme.

Remark 3. *In the above protocol, we used a two message delegatable* $\mathbf{P}$*-certification system with a simple verification algorithm (i.e., the verification algorithm does not depend on the statement). We used the simple verification property since our instantiation based on the message hiding encoding of [KLW14] satisfies this property. However, we remark here that this property is not necessary. In particular, our protocol can be modified as follows to work with any two message delegatable* $\mathbf{P}$*-certification system whose verification algorithm may depend on the statement (i.e.,* $\mathsf{V}_{\mathsf{cert}}$ *takes input* $(1^k, c, \mathsf{CRS}, q, \pi)$*): Simply replace the* $\mathcal{WISSP}$ *in Phase 5, with a* WIUA *to prove that either* $x \in L$ *or there exist* π, CRS, ρ_4, q, ρ_2*, such that,* CRS *is committed to in Phase 4 using random coins* ρ_4*,* q *is committed to in Phase 2 using random coins* ρ_2*, and* $\mathsf{V}_{\mathsf{cert}}(1^k, D, \mathsf{CRS}, q, \pi) = 1$*. The proofs of soundness and zero-knowledge property follow essentially the same.*

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. 2013.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, volume 0, pages 106–115, 2001.

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, pages 345–355, Washington, DC, USA, 2002. IEEE Computer Society.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.

[BGL+14] Bitansky, Garg, Lin, Pass, and Telang. Succint randomized encodings and obfuscations. Manuscript (subsuming an early version appearing as Succint Garbling Schemes and Applications [Lin-Pass, Eprint Report 2014/766]), 2014.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 1992.

[BHR12a] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 134–153, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.

[BHR12b] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004.

[BP12] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, pages 223–232, 2012.

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/`.

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CHJV14] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769, 2014.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.

[CLP13a] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

[CLP13b] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from p-certificates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 50–59, 2013.

[COP+14] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, Muthuramakrishnan Venkitasubramaniam, and Ivan Visconti. 4-round resettably-sound zero knowledge. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 192–216, 2014.

[COPV13] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettability from one-way functions. 2013.

[CPS13] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 231–240, 2013.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.

[DS98] Cynthia Dwork and Amit Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In *CRYPTO*, pages 177–190, 1998.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.

[GGHW13] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Technical report, Cryptology ePrint Archive, Report 2013/860, 2013. 6, 2013.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GJ10] Vipul Goyal and Abhishek Jain. On the round complexity of covert computation. In *STOC*, pages 191–200, 2010.

[GJO+12] Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Concurrent zero knowledge in the bounded player model. Cryptology ePrint Archive, Report 2012/279, 2012. http://eprint.iacr.org/.

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/`.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

[Gol02] Oded Goldreich. Concurrent zero-knowledge with timing, revisited. In *STOC*, pages 332–340, 2002.

[Goy13] Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 221–230, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GS12] Divya Gupta and Amit Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. Cryptology ePrint Archive, Report 2012/572, 2012. `http://eprint.iacr.org/`.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1998.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Redondo Beach, California, USA, November 12–14, 2000. IEEE Computer Society Press.

[IPS14] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. Cryptology ePrint Archive, Report 2014/942, 2014. `http://eprint.iacr.org/`.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In *CRYPTO*, pages 311–324, 1995.

[KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/`.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in polyloalgorithm rounds. In *STOC*, pages 560–569, 2001.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS*, pages 484–492, 1998.

[Lin03] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4:151–158, 1991.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, New York, NY, USA, 2004. ACM.

[Pop63] Karl Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge.* Routledge, 1963.

[PPS13] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for np. Cryptology ePrint Archive, Report 2013/754, 2013. `http://eprint.iacr.org/`.

[PR03a] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–, 2003.

[PR03b] Rafael Pass and Alon Rosen. How to simulate using a computer virus. Unpublished manuscript, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, pages 533–542, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[PRT11] Rafael Pass, Alon Rosen, and Wei-Lung Dustin Tseng. Public-coin parallel zero-knowledge for np. *J. Cryptology*, 2011.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, pages 500–517, 2014.

[PTV12] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent zero-knowledge, revisited. Journal of Cryptology, 2012.

[PV10] Rafael Pass and Muthuramakrishan Venkitasubramaniam. Private coins versus public coins in zero-knowledge proofs. To appear in TCC 2010, 2010.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *Eurocrypt*, pages 415–432, 1999.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO*, pages 451–468, 2000.