Constant-Round Concurrent Zero-knowledge from Indistinguishability Obfuscation

Kai-Min Chung[1], Huijia Lin[2], Rafael Pass[3]

[1] Academia Sinica
kmchung@iis.sinica.edu.tw
[2] University of California, Santa Barbara
rachel.lin@cs.ucsb.edu
[3] Cornell University
rafael@cs.cornell.edu

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).

1 Introduction

Zero-knowledge ($\mathcal{ZK}$) interactive proofs [30] 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 [24], 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 DNS Dwork, Naor and Sahai (which relied on "timing assumptions"), various other concurrent $\mathcal{ZK}$ protocols have been obtained based on different set-up assumptions (e.g., [25, 22]), or in alternative models (e.g., super-polynomial-time simulation [44, 53]).

In the standard model, without set-up assumptions (the focus of our work), Canetti, Kilian, Petrank and Rosen [14] (building on earlier works by [38, 58]) 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 [56] 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) [37] to $\tilde{O}(\log^2 n)$ round. More recent work by Prabhakaran, Rosen and Sahai [55] 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 [52] 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 [33] (and informally discussed in [16]) under various "extractability" assumptions of the knowledge-of-exponent type [21, 34, 8], 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., [24, 44]), 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 [16] 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 $\mathbf{P}$-*certificates*, roughly speaking, succinct non-interactive arguments for languages in $\mathbf{P}$. An issue with their approach, however, is we only have candidate constructions of $\mathbf{P}$-certificates that are sound against *uniform* polynomial-time attackers (as opposed to non-uniform ones), and the protocol of [16] inherits the soundness property of the underlying $\mathbf{P}$-certificate. Additionally, whereas the assumption that a particular proof system is a $\mathbf{P}$-certificates is a falsifiable assumption [54, 42], it is unclear whether the existence of $\mathbf{P}$-certificates itself can be based on some more natural hardness assumptions.

A very recent elegant work by Pandey, Prabhakaran and Sahai [43] 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) $\mathbf{P}/poly$ [6, 11, 1]. Whereas the assumption that a particular scheme is a **diO** is an "extractability" assumption (similar in flavor to knowledge-of-exponent type [21, 34, 8] assumptions), the intriguing part of the scheme of Pandey et al [43] 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 [12, 27, 36]; unless we restrict 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**)* [6, 26]. Following the groundbreaking work of Garg et al [26], there are now several candidate constructions of **iO** that can be based on hardness assumptions on (approximate) multilinear maps [51, 29].

Theorem 1. *Assume the existence of indistinguishability obfuscation for* $\mathbf{P}/poly$ *(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:

1. We first observe that a warm-up case considered in [16]—which shows the existence of constant-round concurrent zero-knowledge based on, so-called, *unique* $\mathbf{P}$*-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).
2. 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) $\mathbf{P}/poly$ [12], and as such this preliminary step already implies the result of [43] in a modular way (but with worse concrete round complexity).
3. We next show how to use fully homomorphic encryption (FHE) [57, 28] and **iO** to modify the protocol to handle also *two-round* unique **P** certificates. Two-round **P**-certificates are a generalization of **P**-certificates in the CRS model, where we allow the CRS (i.e., the "first message" from the verifier to the prover) to depend on the statement to be proven.

4. We finally leverage recent results on delegation of computation based on **iO** from [9, 13, 39] and show that the beautiful scheme of Koppula, Lewko and Waters [39] can be modified into a two-message unique **P**-certificate (also with non-uniform soundness).
   More precisely, we show that any "succint" *message hiding encoding* [39], which is a relaxed version of a "succint" randomized encoding [35, 9], together with injective one-way functions yields a two-round unique **P**-certificate. [39] shows how to construct succint message-hiding encodings based on **iO** and injective PRGs.

The above steps show how to obtain constant-round concurrent $\mathcal{ZK}$ based on collision-resistant hashfunctions, **iO** for $\mathbf{P}/poly$, one-way permutations, and FHE. We finally observe that the message-hiding encoding of [39] has a particular nice structure that enables us to refrain from using FHE in the final protocol, thus reaching our final theorem.

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 [16], which we start by recalling (closely following the description in [16]). The starting point of the construction of [16] is the construction is Barak's [2] non-black-box zero-knowledge argument for NP. Below, we briefly recall the ideas behind his protocol (following a slight variant of this protocol due to [47]).

Barak's protocol. Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\mathrm{poly}(n)}$, the Prover $\mathbf{P}$ 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 $M(\sigma) = r$.[4]

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) [41, 4] 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.

[4] 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 $h(m)$.

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 [47, 50], but there is a serious issue: if the verifier "nests" its concurrent 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 [16]. As shown in [16], 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 prove (using a WIUA again) 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 $\boldsymbol{m}$ 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 $\mathbf{P}$-certificate is unique to argue that the sequence λ is uniquely defined. (Technically, we here need to rely on a $\mathbf{P}$-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 $\mathbf{P}$-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 $\boldsymbol{\pi}$ 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 "left" messages will later be certificates provided by S_2.

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.[5]) 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 $\mathbf{P}$-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 $\mathbf{P}$-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

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

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 [16, 15]; 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 [16] 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.

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 can be 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.[6] 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.

[6] 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.

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.[7] 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 Two-round Unique 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. (On the other hand, while we allow the CRS to depend on the statement q, we require the length of the CRS to the independent of the *lenght* of q.) In essence, we are considering *two-round* publicly-verifiable delegation protocols. We refer to such schemes as *two-round* **P**-*certificates*. We now generalize the above approach to work with unique two-round **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 send an FHE encryption $\hat{\alpha}$ of random coins α needed to run the CRS generation. (Using PRGs, we may assume wlog that the lenght of α is n.)
- 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 commitment to an FHE encryption $\hat{\rho}$ obtained by running the CRS-generation procedure (using the appropriate FHE operations) on the ciphertext $\hat{\alpha}$. (That is, $\hat{\rho}$ is an encryption of the CRS ρ obtained by running the CRS generation algorithm with random coins α.)
- Next, the Verifier sends an indistinguishability obfuscation $\tilde{\Pi} = \mathbf{iO}(\Pi)$ of a program Π that on input a decommitment $(\hat{\rho}, r')$ to c_3 decrypts $\hat{\rho}$ (using the FHE secret key) into a CRS ρ and outputs it. (The reason that the Verifier cannot simply decrypt $\hat{\rho}$ for the Prover is that $\hat{\rho}$ cannot be sent to the Verifier in the clear; recall that the honest prover will never compute any such ciphertext, 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.
- 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 (without any

[7] 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.

knowledge of the random coins α used to generate ρ), 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.

Realizing Unique Two-Round P-Certificates. We finally leverage recent results on delegation of computation based on **iO** from [9, 13, 39] and show that the beautiful scheme of Koppula, Lewko and Waters [39] can be massaged (and slightly modified) into a two-message unique **P**-certificate. More precisely, we show how to use the notion of a "succint" *message hiding encoding* [39]—a relaxed version of a "succint" randomized encoding [35, 9]—together with injective one-way functions to construct a two-round unique **P**-certificate. [39] shows how to construct succint message-hiding encodings (in fact, even succint randomized encodings) based on **iO** for $\mathbf{P}/poly$ and injective PRGs.

Let is point out that, just as [16], 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 rely on **iO** for $\mathbf{P}/poly$ secure against $T(\cdot)$-size circuits.

Removing the use of FHE. In a final step, we note that by relying on specific nice properties of the message-hiding encoding of [39], we obtain a two-round **P**-certificate satisfying a desirable property: Only a "small" part of the CRS generation procedure relies on secret coins. More precisely, the CRS generation procedure proceeds in three steps: 1) first, secret coins are used 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 proportional to the length of q, but this work only requires public information.

We refer to such schemes as *deletegable* **P***-certificates in the CRS model* and note that if we rely on such a scheme, then we can dispense the need for FHE in our final protocol, as the computation of the disgest can be directly delegated to the prover without the need of FHE.

This completes the informal description of our protocol and its proof of security. In our formal description of the final protocol, for simplicity, we directly present a solution using such delegetable **P**-certificates (without going through the construction using FHE). 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 [16]) 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.

Other Related Works. Since the work of Barak [2], non-black-box simulation techniques have been used in several other contexts: Non-malleability [3, 45, 48,

49], concurrent secure computation [40, 46, 45, 7], resettable-soundness [5, 23, 10, 18, 20, 17], covert secure computation [32] and more. We believe our techniques may yield improved constructions also in these settings.

We also mention recent work of [15, 31] 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 [16] (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**.

Organization. In Section 2, we define unique two-message **P**-certificates, and the property of delegatable CRS generation. We show how to instantiate them using message hiding encoding of [39] in the full version of the paper [19]. In Section 3, we present our constant-round concurrent $\mathcal{ZK}$ protocol, and its simulator. We refer the reader to the full version [19] for preliminaries and full proof of our protocol.

2 Two-Message P-certificates

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

Definition 1 (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: *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 2 (Selective Strong Soundness) *We say that a* **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 (e.g.,* $T(n) = n^{\log\log\log n}$*)* $T(k) \in k^{\omega(1)}$ *and some "nice" super-constant function (e.g.,* $C(k) = \log\log\log n$*)* $C(\cdot) \in \omega(1)$ *such that for every probabilistic algorithm* P^*

with running-time bounded by $T(\cdot)$, there exists a negligible function μ, such that, for every $k \in N$, $c \leq C(k)$,

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

Definition 3 (Uniqueness) *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* $\mathsf{V}_{\mathsf{cert}}(1^k, c, \mathsf{CRS}, q, \pi) = 1$.

Definition 4 (Delegatable CRS Generation) *We say that a (two-message)* $\mathbf{P}$*-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 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 $\mathbf{P}$-certificates: We say that the verification algorithm of a $\mathbf{P}$-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.

Instantiation of P-certificates. In the full version of the paper [19], we show that unique two-message $\mathbf{P}$ certificates can be constructed from any "message hiding encoding scheme" [39] and injective one-way functions. Furthermore, we show that the $\mathbf{P}$ certificates instantiated using the specific message hiding encoding of [39] has delegatable CRS generation and a simple verification procedure.

3 Our Protocol

Our constant-round concurrent $\mathcal{ZK}$ protocol relies on the following primitives:

1. 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 in the full version of the paper [19] for more details.
2. A strong (two-message) $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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 w.l.o.g., since our instantiation based on the message hiding encoding of [39] satisfies this property. But this is not necessary. See Remark 3 in the full version of the paper [19] on how to avoid using this property.
3. 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 [4], 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 in the full version of the paper [19] for more discussion.

4. An indistinguishability obfuscator $i\mathcal{O}$ for circuits.
5. 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 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 satisfies

- USE OF EMULATOR: q is 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* $\mathcal{O}^n_{Vcert}$ *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* $\tilde{S}_1^{\mathcal{O}_{Vcert}}(1^n, j, s) = r$*, 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 ρ_{Setup}, 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} hard-wired 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 1.

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.

Fig. 1. 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, $\mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}, \pi) = 1$.
V accepts if the proof is accepting.

NOTE: *In simulation, the simulator computes proof* $\pi \stackrel{\$}{\leftarrow} \mathsf{P}_{\mathsf{cert}}(1^k, D, q, \mathsf{CRS})$*, and succeed in the final proof by using* π *and* CRS, ρ_4 *generated in the last phase as "trapdoor" witness.*

Theorem 1 *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. In the next subsection, we describe the concurrent zero knowledge simulator. The analysis of the simulator and the proof of concurent $\mathcal{ZK}$ property, as well as the soundness proof, are provided in the full version of the paper [19].

3.1 Contruction of the Simulator

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 $\boldsymbol{\pi}$; 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 $\boldsymbol{\pi}$ from the $\mathbf{P}$-certificate oracle.

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 **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 $\mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}_i, \pi_i) = 1$ w.r.t. CRS_i which is the committed value in Phase 4 of session i.

PROCEDURE OF SIMULATOR S_2: $S_2(1^n, x, M, s, \ell)$ 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, $\pi_i \xleftarrow{\$} \mathsf{P}_{\mathsf{cert}}(1^n, D, q_i, \mathsf{CRS}_i)$ (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 $\mathsf{V}_{\mathsf{cert}}(1^n, \mathsf{CRS}_i, \pi_i) = 1$, 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.

PROCEDURE OF THE FINAL SIMULATOR S: 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.)

Due to the lack of space, the analysis of the simulator, including its running time, and the correctness of its output distribution is provided in the full version of the paper [19], which also contains the soundness proof.

References

1. P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. 2013.
2. B. Barak. How to go beyond the black-box simulation barrier. In *FOCS*, volume 0, pages 106–115, 2001.
3. B. 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.
4. B. Barak and O. Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.
5. B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.
6. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.
7. B. Barak and A. Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.
8. M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004.
9. 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.
10. N. Bitansky and O. Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, pages 223–232, 2012.
11. E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.
12. E. Boyle and R. Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/`.
13. R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769, 2014.
14. R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.
15. R. Canetti, H. Lin, and O. Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

16. K. Chung, H. Lin, and R. 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.
17. K. Chung, R. Ostrovsky, R. Pass, M. Venkitasubramaniam, and I. 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.
18. K. Chung, R. Pass, and K. 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.
19. K.-M. Chung, H. Lin, and R. Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/991, 2014. `http://eprint.iacr.org/`.
20. K.-M. Chung, R. Ostrovsky, R. Pass, and I. Visconti. Simultaneous resettability from one-way functions. 2013.
21. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.
22. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.
23. Y. Deng, V. Goyal, and A. Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.
24. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.
25. C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In *CRYPTO*, pages 177–190, 1998.
26. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.
27. S. Garg, C. Gentry, S. Halevi, and D. 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.
28. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.
29. C. Gentry, A. Lewko, A. Sahai, and B. Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/`.
30. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
31. V. Goyal. Non-black-box simulation in the fully concurrent setting. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 221–230, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
32. V. Goyal and A. Jain. On the round complexity of covert computation. In *STOC*, pages 191–200, 2010.
33. D. Gupta and A. Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. Cryptology ePrint Archive, Report 2012/572, 2012. `http://eprint.iacr.org/`.
34. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1998.
35. Y. Ishai and E. 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, Nov. 12–14, 2000. IEEE Computer Society Press.
36. Y. Ishai, O. Pandey, and A. Sahai. Public-coin differing-inputs obfuscation and its applications. Cryptology ePrint Archive, Report 2014/942, 2014. `http://eprint.iacr.org/`.
37. J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *STOC*, pages 560–569, 2001.
38. J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS*, pages 484–492, 1998.
39. V. Koppula, A. B. Lewko, and B. Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/`.
40. Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.
41. S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
42. M. Naor. On cryptographic assumptions and challenges. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.
43. O. Pandey, M. Prabhakaran, and A. 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/`.
44. R. Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.
45. R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, New York, NY, USA, 2004. ACM.
46. R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–, 2003.
47. R. Pass and A. Rosen. How to simulate using a computer virus. Unpublished manuscript, 2003.
48. R. Pass and A. Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.
49. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, pages 533–542, 2005.
50. R. Pass, A. Rosen, and W.-L. D. Tseng. Public-coin parallel zero-knowledge for np. *J. Cryptology*, 2011.
51. R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, pages 500–517, 2014.
52. R. Pass, W.-L. D. Tseng, and M. Venkitasubramaniam. Concurrent zero-knowledge, revisited. Journal of Cryptology, 2012.
53. R. Pass and M. Venkitasubramaniam. Private coins versus public coins in zero-knowledge proofs. To appear in TCC 2010, 2010.
54. K. Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge, 1963.
55. M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.
56. R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *Eurocrypt*, pages 415–432, 1999.
57. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms, 1978.
58. A. Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO*, pages 451–468, 2000.