Constant-Round Concurrent Zero Knowledge From Falsifiable Assumptions

Kai-Min Chung[*] Huijia Lin[†] Rafael Pass[‡]

Abstract

We present a constant-round concurrent zero-knowledge protocol for **NP**. Our protocol is sound against uniform polynomial-time attackers, and relies on the existence of families of collision-resistant hash functions, and a new (but in our eyes, natural) falsifiable intractability assumption: Roughly speaking, that Micali's non-interactive CS-proofs are sound for languages in **P**.

[*] Cornell University, Email: `chung@cs.cornell.edu`

[†] MIT and Boston University, Email: `huijia@csail.mit.edu`.

[‡] Cornell University, Email: `rafael@cs.cornell.edu`. Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, 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 [Pas03b, 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. Somewhat surprisingly, the simulator strategy of KP is "oblivious"—the "rewinding schedule" of the simulator ignores how the malicious verifier schedules its messages. The key insight behind this oblivious simulation technique is that a single "rewinding" may be helpful for simulating multiple sessions; in essence, KP performs an amortized analysis, which improves the round-complexity. (As we shall see shortly, such an amortized analysis will play an important role also in this work.) 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. A major breakthrough towards resolving this question came with the work of Barak [Bar01], demonstrating a new non-black-box simulation technique that seemed amenable for constructing constant-round protocols that are resilient to concurrent attacks. Indeed, Barak demonstrated a constant-round *bounded-concurrent* argument for **NP** based on the existence of collision-resistant hash-functions; bounded-concurrency here means that for every *a-priori* polynomial bound m on the number of concurrent executions, there exists a protocol (which depends on m) that remains zero-knowledge as long as the number of concurrent execution does not exceed m. (In particular, in the protocol of Barak, the message length of the protocol grows linearly with the *a-priori* bound m on the number of concurrent executions.)

But a decade later, the question of whether "full" (i.e., unbounded) concurrent zero-knowledge is achievable in a constant number of rounds is still wide open.

1.1 Our Results

In this work, we present new falsifiable intractability assumptions, which in our eyes are both natural and reasonable, under which constant-round concurrent zero-knowledge is achievable.

P-certificates We consider an analogue of Micali's non-interactive CS-proofs [Mic00] for languages in **P**. 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). That is, a **P**-certificate allows every deterministic polynomial-time computation to be certified using a "short" certificate (of *a-priori* bounded polynomial length) that can be "quickly" verified (in *a-priori* bounded polynomial-time).

The soundness condition of a **P**-certificate system states that no *uniform* polynomial-time algorithm can output an accepting certificate for any false statement. For our application we will require a slightly stronger soundness condition: soundness needs to hold even against $T(\cdot)$-time attackers attempting to prove the validity also of $T(\cdot)$-time computations, where $T(\cdot)$ is some "nice" (slightly) super-polynomial function (e.g., $T(n) = n^{\log\log\log n}$). We refer to such proof systems as *strong* **P**-*certificates.* Since we consider only languages in **P**, we may also consider *statistically-sound* (resp *statistically-sound strong*) **P**-certificates, where soundness holds also with respect to unbounded attackers restricted to selecting statements of polynomial (resp. $T(\cdot)$) length. (Note that considering soundness against non-uniform efficient-time attackers is equivalent to statistical soundness, since if an accepting proof of a false statement exists, a non-uniform efficient attacker can simply get it as non-uniform advice.

On the Existence of P-certificates A candidate construction of a (computationally-sound) **P**-certificate system comes from Micali's CS-proofs [Mic00]. These constructs provide short certificates even for all of **NEXP**. However, since we here restrict to certificates only for **P**, the assumption that these constructions are sound (resp. strongly sound) **P**-certificates is *falsifiable* [Pop63, Nao03]: Roughly speaking, we can efficiently test if an attacker outputs a valid proof of an incorrect statement, since whether a statement is correct or not can be checked in deterministic polynomial time. Formalizing this intuition turns out to be somewhat subtle: in general, whether an attacker breaks soundness of a strong **P**-certificate system, or even just a **P**-certificate system, may not be efficiently testable since there is no *a-priori polynomial* upper-bound on the running-time of the machine M selected by the attacker. At first one may think that this issue can be easily resolved by asking the prover to provide an upper-bound on the running-time of M in unary; this certainly makes the soundness condition falsifiable, but such certificates are no longer "short". We overcome this issue by relying on the fact that Micali's construction satisfies an additional (and very natural) property, which we refer to as *time-representation invariance*—namely, that whether the verifier accepts a proof of a statement x does not depend on how the time-bound (i.e., the upper bound on the running time of M) is represented. For a time-representation invariant **P**-certificate, it suffices to define soundness for the the case that the attacker specifies the time-bound in unary; by the time-representation invariance condition, this implies soundness also for other (more efficient) representations. Thus, assuming that the soundness condition of a time-representation invariant **P**-certificate holds is a falsifiable assumption, yet "short" certificates can still be generated by using more efficient representations of the running-time bound.[1]

In our eyes, on a qualitatively level, the assumption that Micali's CS-proofs yield strong **P**-certificates is not very different from the assumption that e.g., the Full Domain Hash [BR93] or Schnorr [Sch91] signature schemes are existentially unforgeable: 1) whether an attacker succeeds can

[1] In contrast, as shown by Gentry and Wichs [GW11], (under reasonable complexity theoretic assumptions) non-interactive CS-proofs for **NP** cannot be based on any falsifiable assumption using a black-box proof of security.

be efficiently checked, 2) no attacks are currently known, and 3) the "design-principles" underlying the construction rely on similar intuitions.

Finally, note that the assumption that statistically-sound strong **P**-certificates exists is implied by the assumption that 1) $\mathbf{DTIME}(n^{\omega(1)}) \subseteq \mathbf{NP}$ and 2) **NP** proofs for statements in $\mathbf{DTIME}(t)$ can be found in time polynomial in t [BLV06]. In essence, the assumption says that non-determinism can slightly speed-up computation, and that the non-deterministic choices can be found efficiently, where efficiency is measured in terms of the original deterministic computation. Although we have no real intuition for whether this assumption is true or false[2], it seems beyond current techniques to contradict it. (As far as we know, at this point, there is no substantial evidence that even $\mathbf{SUBEXP} \not\subseteq \mathbf{NP}$.)

From P-certificates to O(1)-round Concurrent $\mathcal{ZK}$ Our main theorem is the following.

Theorem. *Assume the existence of families of collision-resistant hash-functions secure against polynomial-size cirucuits, and the existence of a strong* **P***-certificate system (resp. a statistically-sound strong* **P***-certificate system). Then there exists a constant-round concurrent zero-knowledge argument for* **NP** *with uniform soundness (resp. non-uniform soundness). Furthermore, the protocol is public-coin and its communication complexity depends only on the security parameter (but not on the length of the statement proved).*

Our protocol is a variant of Barak's [Bar01] non-black-box zero-knowledge argument for **NP**. As mentioned above, Barak's original protocol already handles *bounded-concurrent* composition; that is, it remains secure under an *a priori* bounded number of concurrent executions. In contrast, our protocol handles an unbounded number of executions, but relies on (seemingly) stronger assumptions.

Let us briefly remark that from a theoretical point of view, we find the notion of uniform soundness of interactive arguments as well-motivated as the one of non-uniform soundness; see e.g., [Gol93] for further discussion. From a practical point of view (and as is often the case), an asymptotic treatment of soundness is not needed for our results, even in the uniform setting: our soundness proof is a constructive black-box reduction that (assuming the existence of families of collision-resistant hash-functions), transforms any attacker that breaks soundness of our concurrent $\mathcal{ZK}$ protocol on a *single* security parameter 1^n into an attacker that breaks the the soundness of the **P**-certificate systems with comparable probability on the *same* security parameter 1^n, with only a "small" polynomial overhead. In particular, if some attacker manages to break the soundness of a particular instantiation of our protocol using e.g., Micali's CS-proof for languages in **P** implemented using some specific hash function (e.g., SHA-256), then this attacker can be used to break this *particular* implementation of CS-proofs.

1.2 Outline of Our Techniques

We provide a detailed outline of our techniques. We warn the reader that this outline is quite technical and assumes the reader is relatively familiar with Barak's non-black-box simulation technique.

Let us start by very briefly recalling the idea behind Barak's protocol (following a slight variant of this protocol due to [PR03b]). Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\text{poly}(n)}$, the

[2] As far as we know, the only evidence against it is that it contradicts very strong forms of derandomization assumptions [BLV06, BOV07].

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 $M(\sigma) = r$.[3]

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 $m = poly(n)$ 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 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).

$\mathbf{P}$-certificates to The Rescue Our principal idea is to use $\mathbf{P}$-certificates to overcome the above-mentioned blow-up in the running time. On a very high-level, the idea is that once the simulator S has generated a $\mathbf{P}$-certificate π to certify some partial computation performed by S in a particular session i, then the same certificate may be reused (without any additional "cost") to certify the same computation also in other sessions $i' \neq i$. In essence, by reusing the same $\mathbf{P}$-certificates, we can amortize the cost of generating them and may then generate WIUA's about WIUA's etc., without blowing-up the running time of the simulator. Let us briefly mention how the two salient features of $\mathbf{P}$-certificates, namely "non-interactivity" and "succinctness", are used: Without non-interactivity, the same certificate cannot be reused in multiple sessions, and without succinctness, we do not gain anything by reusing a proof, since just reading the proof may be more expensive than verifying the statement from "scratch".

Implementing the above high-level idea, however, is quite non-trivial. Below, we outline our actual implementation. We proceed in three steps:

1. We first present a protocol that only achieves bounded-concurrent $\mathcal{ZK}$, using $\mathbf{P}$-certificates,

[3] 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 $h(m)$.

2. We next show how this bounded-concurrent protocol can be slightly modified to become a (fully) concurrent $\mathcal{ZK}$ protocol assuming the existence of so-called *unique* **P**-*certificates*—**P**-certificates having the property that for every true statement, there exists a *single* accepting certificate.

3. In the final step, we show how to eliminate the need for uniqueness, by generating **P**-certificates about the generation of **P**-certificates etc., in a tree-like fashion.

Step 1: Bounded Concurrency Using P-certificates In this first step, we present a (somewhat convoluted) protocol using strong **P**-certificates that achieves $m(\cdot)$-bounded concurrency (using an even more convoluted simulation). As mentioned, Barak's original protocol could already be modified to handle bounded concurrency, without the use of **P**-certificates; but as we shall see shortly, our protocol can later be modified to handle full concurrency.

The protocol proceeds just as Barak's protocol in Stage 1 except that the verifier now sends a string $r \in \{0,1\}^{2m(n)n^2}$ (instead of length $2n$). 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 $\sigma \in \{0,1\}^{m(n)n^2}$ 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 *"short"* string $\sigma \in \{0,1\}^{m(n)n}$, and
- **prediction correctness:** there exists a **P**-certificate π of length n demonstrating that $M_1(\sigma) = r$.

(Note that the only reason we still need to use a *universal* argument is that there is no *a-priori* upper-bound on the length of the program M_1; the use of the **P**-certificate takes care of the fact that there is no *a-priori* upper-bound on the running-time of M_1, though.) Soundness follows using essentially the same approach as above, except that we now also rely on the strong soundness of the **P**-certificate; since there is no a-priori upper-bound on neither the length nor the running-time of M_1, we need to put a cap on both using a (slightly) super-polynomial function, and thus to guarantee soundness of the concurrent zero-knowledge protocol, we need the **P**-certificate to satisfy *strong* soundness.

Let us turn to (bounded-concurrent) 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 "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^*.

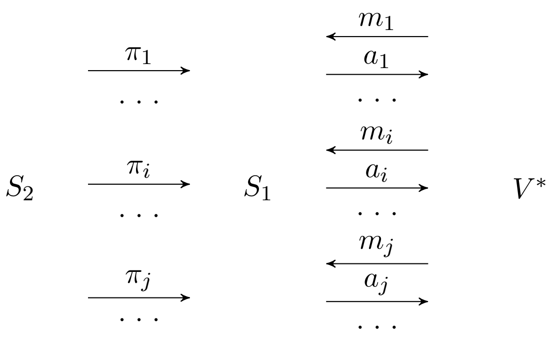

Figure 1: Simulation using **P**-certificates.

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.[4] 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 message 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. Additionally, note that if there are $m(n)$ sessions, the length of σ is at most $O(m(n)n) \ll m(n)n^2$—for each of the $m(n)$ sessions, and for each round of the constant number of rounds in each session, we need to store a pair (j, π) where π is of length n; therefore, the simulation always succeeds without getting "stuck".

Finally, indistinguishability of this simulation, roughly speaking, should follow from the hiding property of the commitment in Stage 1, and the WI property of the WIUA in Stage 2. Or does it? Note that since S_1 is committing to its own code (including its randomness), it is committing to a message that depends on the randomness used for the commitment. (In the language of [BCPT12], this constitutes a randomness-dependent message (RDM) attack on the commitment scheme.) This circularity can be easily overcome (as in [PRT11]) by simply not committing to the randomness of $\tilde{S}_1$, and instead providing it as an additional input to $\tilde{S}_1$ that may be incorporated in σ; without loss of generality, we may assume that the randomness is "short" since S_1 can always use a PRG to expand it. But the same circularity arises also in the WIUA, and here σ, which contains the seed used to generate the randomness of S_1, needs to be an input. To overcome it, we here require S_1 to use a *forward-secure* PRG [BY03] to expand its randomness; roughly speaking, a forward-secure

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

PRG ensures that "earlier" chunks of the output of the PRG are indistinguishable from random, even if a seed generating the "later" ones is revealed. We next have S_1 use a new chunk of the output of the PRG to generate each new message in the interaction, but uses these chunk in *reverse order* (i.e., in step 1, the last chunk of the output of the PRG is used, etc.); this means that we can give proofs about "earlier" computations of S_1 (which requires knowing a seeds expanding the randomness used in the computation) while still guaranteeing indistinguishability of "later" messages.[5]

Step 2: Full Concurrency using Unique P-certificates The reason that the above approach only yields a bounded concurrent zero-knowledge protocol is that for each new session i, we require S_2 to provide S_1 with new certificates, which thus grows the length of σ. If we could somehow guarantee that these certificates are *determined* by the statement proved in the WIUA, then soundness would hold even if σ is long. Let's first sketch how to do this when assuming the existence of *unique* strong **P**-certificates—that is, **P**-certificates having the property that for each true statement x, there exists a single proof π that is accepted. (We are not aware of any candidates for unique **P**-certificates, but using them serves as a simpler warm-up for our actual protocol.) We simply modify the input certification and prediction correction conditions in the WIUA to be the following:

- **input certification:** there exists a vector $\lambda = ((1, \pi_1), (2, \pi_2), \ldots)$ and a vector of messages $\vec{m}$ such that π_i certifies that $M_1(\lambda_{<j})$ output 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. Zero-knowledge, on the other hand, can be shown in exactly the same way as above (using S_1, S_2), but we can now handle also unbounded concurrency (since there is no longer a restriction on the length of the input λ).

Step 3: Full Concurrency Using (Plain) P-certificates Let us finally see how to implement the above idea while using "plain" (i.e., non-unique) **P**-certificates. The above protocol is no longer sounds since we cannot guarantee that the proofs π_j are unique, and thus the messages m_j may not be unique either, which may make it possible for an attacker to pass the "prediction correctness" condition (without knowing the code of V^*) and thus break soundness. A natural idea would thus be to ask the prover to commit to a machine M_2 (which in the simulation will be a variant of S_2) that produces the certificates π_j, and then require the prover to provide a "second-level" certificate that the "first-level" certificates were generated (deterministically) by running M_2. But have we really gained anything? Now, to perform the simulation, we need to provide the second-level certificates as input to both M_1 and M_2; however, for these second-level certificates, we have no guarantees that they were deterministically generates and again there is no *a-prior* upper bound on the number of such certificates, so it seems we haven't really gained anything.

[5] Although the language of forward-security was not used, it was noticed in [PR03b] that GGM's pseudo-random function [GGM86] could be used to remove circularity in situations as above. A related trick is used in the contemporary work of [CLP12].

Our main observation is that a *single* ”second-level” certificate can be used to to certify the (deterministic generation) of n ”first-level”certificates. And a sequence of n “second-level” certificates can be certified by a single “third-level” certificate, etc. At each level, there will be less than n certificates that are *not* certified by a higher-level certificate; we refer to these as “dangling” certificates. See Figure 2 for an illustration of the tree structure, and certified and dangling certificates.

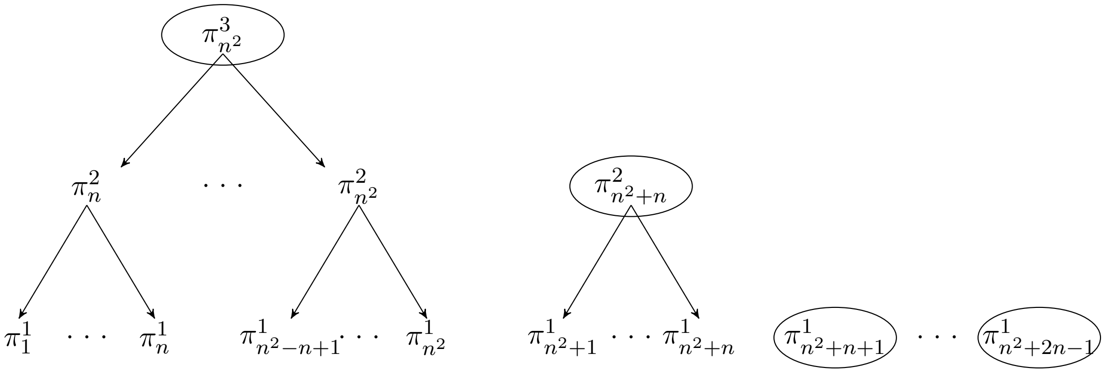

Figure 2: An illustration of the tree structure for generating **P**-certificates. Nodes that are not circled are “certified” certificates; nodes that are circled are “dangling” certificates.

Note that since the number of messages in the interaction with V^* is polynomially bounded, we only have a polynomial-number of level-1 certificates, and thus, the above higher-level certification process does not go beyond a constant number of levels (at each level we need a factor of n less certificates). Finally, note that the total number of “dangling” (uncertified) certificates is bounded by the number of levels times n (and is thus bounded by, say, n^2.) This means that all the dangling certificates may be provided as a “short” input σ to the committed program, and all the certified certificates can be provided as a “long” (but certified deterministically generated) input λ.

Let us explain this idea more closely using only second-level certificates; this still only gives us bounded-concurrency, but we may now handle $O(m(n)n)$ sessions (instead of just $m(n)$). (More generally, if we use k-levels of certification, we can handle $m(n)n^k$ sessions.) We now change Stage 2 of the protocol to require P to use a WIUA to prove that either x is true, or

- **commitment consistency:** c is a commitment to programs M_1, M_2, and
- **input certification:** there exists
  - a vector of ”certified level-1 certificates” $\lambda^1 = ((1, \pi_1), (2, \pi_2), \ldots, (an, \pi_{an}))$,
  - a ”small” number of ”dangling level-1 certificates” $\sigma^1 = (\sigma^1_1, \sigma^1_2, \ldots, \sigma^1_{j'})$, where $j' < n$ and for each $j \leq j'$, $\sigma^1_j \in \{0,1\}^n$,
  - $a \leq m(n)$ level-2 certificates $\sigma^2 = (\sigma^2_n, \sigma^2_{2n}, \ldots, \sigma^2_{an})$ where for each $j \leq a$, $\sigma^2_{jn} \in \{0,1\}^n$,

  such that,

  - σ^2_{an} certifies that $M_2(\sigma^2_{<an})$ generates the certificates λ^1,

  and
- **prediction correctness:** there exists a **P**-certificate π of length n demonstrating that $M_1(\lambda^1, \sigma^1, \sigma^2) = r$.

Soundness of this protocol follows since the total length of "arbitrary" (not deterministic) input is bounded by $(m(n)+n)n \ll m(n)n^2$. $m(n)n$-bounded concurrent zero-knowledge on the other hand, roughly speaking, follows by letting M_1 be as above (i.e., $\tilde{S}_1$) and M_2 be a variant of the simulator S_2 that outputs all the certificates generated by S_2. We then define a simulator S_3 responsible for generating second-level certificates for S_2, and finally outputs its full view of the interaction. The final simulator S runs S_3 and outputs the view of V^* in the interaction. See Figure 3 for an illustration of the communication patterns of S_1, S_2, S_3 and V^*.

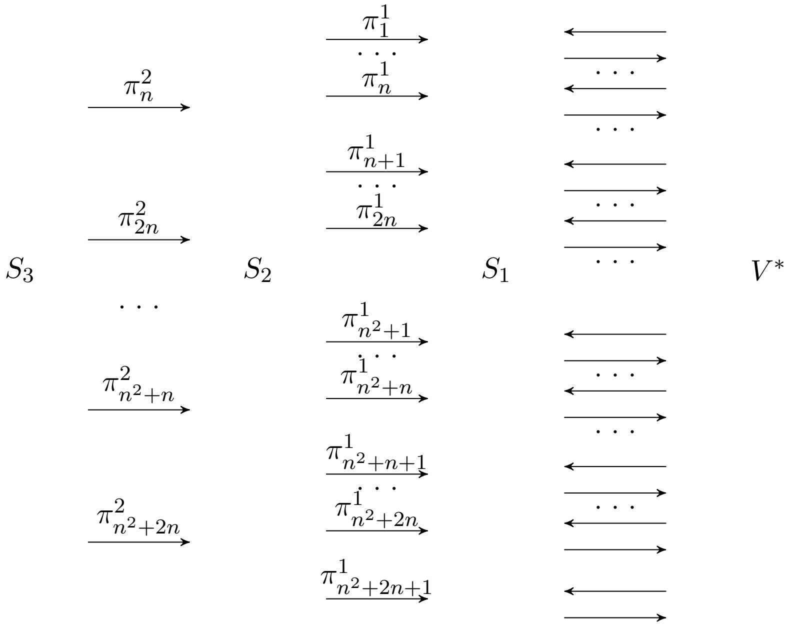

Figure 3: Simulation using second-level **P**-certificates.

Note that as long as there are less than $m(n)n$ message in the interaction with V^*, the number of first-level certificates is bounded by $m(n)n$, and thus we have enough "spots" for second-level certificates (in σ^2) to perform the simulation.

In the final protocol, we instead have the simulator commit to a sequence $M_1, M_2, \ldots$ of machine; roughly speaking, M_1 will be as above, M_2 is responsible for generating first-level certificates (while receiving level $k > 1$ certificates externally), M_3 will be responsible for generating second-level certificates (while receiving level $k > 2$ certificates externally), etc. Note that although there is a (potentially) exponential blow-up in the time needed to generate higher-level certificates, since we only have a constant-number of levels, simulation can be performed in polynomial-time.

1.3 Related Work

We provide a detailed discussion of some other related works:

- As mentioned in the introduction, constant-round concurrent zero-knowledge protocols with *super-polynomial-time* simulators have been constructed in the plain model [Pas03a, PV08]. For the protocol of [Pas03a], the only super-polynomial-time "advantages" needed by the simulator is to find a pre-image $x' = f^{-1}(y)$ to any point y output by the malicious verifier V^*, as long as y actually is in the range of some one-way function f. If we *assume* that the only way for V^* to output some y in the range of f is by applying f to an input x *that it* *explicitly knows*, then the protocol of [Pas03a] is concurrent zero-knowledge. A problem with formalizing this is that V^* may already get some string $y = f(x)$ as its auxiliary input and thus may not know x. As in the literature on "knowledge-of-exponent"-type *extractability assumptions* (see e.g., [Dam91, HT98, BP04b, CD09, BCCT12a, DFH12, GLR11]), this issue can be resolved by having the prover select the one-way function f from a family $\mathcal{F}$ of one-way functions. Now the extractability assumption we need is that for every polynomial-time oracle machine M, there exists some polynomial-time machine M' such that given any $z \in \{0,1\}^*$, and uniformly selected functions $\vec{f} = f_1, \dots f_{\mathrm{poly}(n)} \in \mathcal{F}$, $M^{O(\vec{f})}(1^n, z, \vec{f})$ and $M'(1^n, z, \vec{f})$ generate the same output, where $O(\vec{f})$ is an oracle that inverts the functions in $\vec{f}$. In other words, we are assuming that in the simulation, the simulator together with the verifier can —in polynomial-time—emulate the one-way function inverter used in [Pas03a]. Note that the above extractability assumption is stronger than the typical "knowledge-of-exponent"-type extractability assumptions since we require simultaneous extractability of many images y that are chosen adaptively by the adversary.[6] However, as shown in [Pas03b], any sufficiently length-expanding random oracle satisfies exactly such an extractability assumption—this was used in [Pas03a] to construct a concurrent $\mathcal{ZK}$ protocol in the "non-programmable" random oracle model.

One important difference between the above approach and our work is that we here provide an *explicit* concurrent $\mathcal{ZK}$ simulator. The above-mentioned approach simply *assumes* that such a simulator exists; and, even if the assumption is true, it is not clear, how to find it. In particular, for the purpose of *deniability* (see e.g., [DNS04, Pas03b]) it is not clear whether the approach based on "extractability" assumptions provides sufficient guarantees (unless an explicit simulator strategy is found).

- Barak, Lindell and Vadhan [BLV06] show that under the assumptions that 1) $\mathbf{DTIME}(n^{\omega(1)}) \subseteq \mathbf{NP}$ and 2) $\mathbf{NP}$ proofs for statements in $\mathbf{DTIME}(t)$ can be found in time polynomial in t, 2-round proof exists that are zero-knowledge for uniform verifiers that do not receive any auxiliary input. Their zero-knowledge simulator is non-black-box. As mentioned in the introduction, the above-mentioned assumptions imply the existence of statistical strong **P**-certificates. We note that the protocol of [BLV06] is not known to be concurrent (or even sequential) zero-knowledge, even with respect to uniform malicious verifiers.

- Contemporary work by Canetti, Lin and Paneth [CLP12] constructs a *public-coin* concurrent zero-knowledge protocol using non-black-box simulation techniques[7]. As shown by Pass, Tseng and Wikstrom [PTW11], public-coin concurrent (and in fact even parallel) zero-knowledge protocols require non-black-box simulation, no matter what the round-complexity is. The protocol of [CLP12] is in the "non-programmable" CRS model of [Pas03a] but as showed in [Pas03a] black-box separation of the Goldreich-Krawczyk [GK96] type (and, in particular, the [PTW11] one, falls into this category) extend also to zero-knowledge in the non-programmable CRS model; thus non-black-box simulation is necessary also for their result. In contrast to our protocol, theirs, however, requires $O(\log^{1+\varepsilon} n)$ number of rounds for arbitrarily small constant ε, but instead only relies on the existence of families of collision-resistant hash functions. (Additionally, [CLP12] note that if assuming the existence of a single hash function that is collision-resistant against uniform adversaries, their protocol can

[6] On the other hand, it is weaker that most other usages of extractability in it requires less structure from the function (i.e., only one-wayness).

[7] Our results and theirs were developed in parallel.

be instantiated also in the plain model with uniform soundness.)

On a technical level, both our work and theirs provide methods for overcoming the exponential blow-up in the simulation time when dealing with non-black-box simulations, but the actual details of the methods are very different: [CLP12] increases the round-complexity to tackle this blow-up, and relies on ideas from the literature on concurrent zero-knowledge with *black-box simulation* [RK99, KP01, PRS02]; as a result, their techniques only apply in the context of super-logarithmic round protocols. In contrast, we rely on **P**-certificates to overcome the blow-up and obtain a constant-round protocol. (We also mention that our protocol can be modified in a straight-forward way to achieve non-uniform soundness in the non-programmable CRS model, by using 2-round **P**-certificates (that are sound against non-uniform polynomial-time) and simply having the first message of the **P**-certificate be fixed as the CRS.)

- A recent work by Bitansky, Canetti, Chiessa, Tromer [BCCT12b] present techniques for composing SNARKs (succinct non-interactive arguments of knowledge) for **NP**; roughly speaking, [BCCT12b] shows that if for *some* sufficiently large c, any *non-deterministic* n^c computation can be proved using an "argument of knowledge" of length n that can be verified in time n^2, then for *any* d, every non-deterministic n^d-time computation can be also be proved (using a SNARK of length n that can be verified in time n^2). This is achieved by having the prover first generate a SNARK for each subcomputation of n^c steps, and then for each "chunk" of n SNARKs, having the prover prove that it *knows* SNARKs that are accepted for all these subcomputations, and so on in a tree-like fashion. Finally, the prover only provides the verifier with a "top-level" SNARK that it knows lower-level SNARKs that proves that it knows even lower-level SNARKs etc. This type of proof composition was previously also used by Valiant [Val08]. To prove that this type of composition works it is crucial to work with languages in **NP** (since we are proving statements about the *existence* of some SNARKs); additionally, it is crucial that we are dealing with arguments *of knowledge*—SNARKs of false statements may exists, so to guarantee soundness, the prover needs to show that not only appropriate SNARKs exists, but also that it "knows" them.

  At a superficial level, our simulator strategy also uses a tree of "proofs". However, rather than proving knowledge of lower-level "proofs" etc, in our approach, higher-level **P**-certificates are only used to demonstrate that lower-level **P**-certificates have been deterministically generated. As a consequence, we do not need to certify non-deterministic computations; additionally, we do not need the certificates to satisfy an argument of knowledge property. Indeed, this is what allows us to base **P**-certificates on a falsifiable assumption.

- 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], concurrent secure computation [Lin03, PR03a, Pas04, BS05], covert secure computation [GJ10]. We believe our techniques will yield improved constructions also for these settings; we hope to report on this in future works.

1.4 Acknowledgements

We are very grateful to Ran Canetti, Johan Håstad Omer Paneth, and Alon Rosen for many discussions about concurrent zero-knowledge and non-black-box simulation. We are especially grateful to both Alon Rosen and Omer Paneth for very insightful discussions about how to formalize non-black-box simulations that "commit to their own code"; additionally, as we mention in the paper, several obstacles to using non-black-box simulation to obtain constant-round concurrent

zero-knowledge were noticed in an unpublished manuscript with Alon dating back to 2003 [PR03a]. Thanks a lot!

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 Witness Relations

We recall the definition of a witness relation for a **NP** language [Gol01].

Definition 1 (Witness relation)**.** A *witness relation* for a language $L \in \mathbf{NP}$ is a binary relation R_L that is polynomially bounded, polynomial time recognizable and characterizes L by $L = \{x : \exists y\, s.t.\, (x, y) \in R_L\}$

We say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$. We will also let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e., $R_L(x) = \{y : (x, y) \in L\}$. In the following, we assume a fixed witness relation R_L for each language $L \in \mathbf{NP}$.

2.2 Computational Indistinguishability

The following definition of computational indistinguishability originates in the seminal paper of Goldwasser and Micali [GM84]. Let X be a countable set of strings. A *probability ensemble indexed by* X is a sequence of random variables indexed by X. Namely, any element of $A = \{A_x\}_{x\in X}$ is a random variable indexed by X.

Definition 2 (Indistinguishability)**.** Let X be a countable set. Two ensembles $\{A_{n,x}\}_{n\in N, x\in X}$ and $\{B_{n,x}\}_{n\in N, x\in X}$ are said to be *computationally indistinguishable over* N if for every probabilistic machine D (the distinguisher) whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $n \in N$ and $x \in X$:

$$|\Pr [a \leftarrow A_{n,x} \ : \ D(1^n, x, a) = 1] - \Pr [b \leftarrow B_{n,x} : D(1^n, x, b) = 1]| < \nu(n)$$

2.3 Interactive Proofs and Arguments

We recall the standard definitions of interactive proofs [GMR89] and arguments (a.k.a computationally sound proofs) [BCC88]. In our definition of arguments, we distinguish between uniform soundness, where soundness only needs to hold against a uniform probabilistic polynomial-time algorithms, and non-uniform soundness, where it holds against non-uniform polynomial-time algorithms. Typically, in the literature on zero-knowledge argument, non-uniform soundness is more commonly used (but there are exceptions, see e.g., [BP04a]). We find the uniform model of computation as well-motivated as the non-uniform one; see e.g., [Gol93].

Definition 3 (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 polynomial-time (resp. non-uniform polynomial-time) machines B_1, B_2, the pair (P, V) is called a *uniformly-sound (resp. non-uniformly sound) interactive argument system.*

2.4 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 4 (Witness-indistinguishability)**.** An interactive protocol (P, V) for $L \in \mathbf{NP}$ 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:

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

2.5 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 binding and statistically hiding; we only make use of statistically binding commitments in this paper. Below we sketch the properties of a statistically binding commitment; full definitions can be found in [Gol01].

In statistically binding commitments, the binding property holds against unbounded adversaries, while the hiding property only holds against computationally bounded (non-uniform) adversaries. The statistical-binding property asserts that, with overwhelming probability 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 statistically-binding commitment schemes can be constructed using any one-to-one one-way function (see Section 4.4.1 of [Gol01]). Allowing some minimal interaction (in which the receiver first sends a single random initialization message), statistically-binding commitment schemes can be obtained from any one-way function [Nao91, HILL99].

2.6 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 5 (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 $(M, x, t) \in \{0,1\}^{\mathrm{poly}(n)} \setminus L_{\mathcal{U}}$,
$$\Pr[(P_n^*, V)(1^n, (M, x, t)) = 1] < \nu(n)$$

- *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 $y = (M, x, t) \in \{0,1\}^{\mathrm{poly}(n)}$ if $\Pr[(P_n^*, V)(1^n, y) = 1] > 1/p(n)$ then
$$\Pr_r[\exists w = w_1, \dots w_t \in \mathbf{R}_{\mathcal{U}}(y) \;\; s.t. \;\; \forall i \in [t], \; E_r^{P_n^*}(1^n, y, i) = w_i] > \frac{1}{p'(n)}$$
where $\mathbf{R}_{\mathcal{U}}(y) \overset{\text{def}}{=} \{w : (y, w) \in \mathbf{R}_{\mathcal{U}}\}$ 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 $L_{\mathcal{U}} \cap \{0,1\}^{\mathrm{poly}(n)}$, 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 $\mathrm{poly}(n) \cdot t$ 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 based on the existence of families of collision-resistant hash functions.

2.7 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 6 (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 7 (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 $n \in N$.

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

2.8 Forward Secure PRG

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.[8]

Definition 8 (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

[8] 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.

3 P-certificates

In this section we define the notion of **P**-certificates. On a high-level, **P**-certificates can be viewed as an analogue of Micali's CS-proofs [Mic00], but where we restrict to languages in **P**. As we shall see, by restricting to languages in **P**, we can make the soundness condition of (a restricted class of) **P**-certificates falsifiable.

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); that is, any deterministic polynomial-time computation can be certified using a "short" certificate that can be "quickly" verified.

To formalize this, 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 9. A pair of probabilistic interactive Turing machines, $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$, is a **P**-*certificate system* if there exist polynomials g_P, g_V, ℓ such that the following holds:

- *Efficient Verification:* On input $c \geq 1$, 1^k and a statement $q = (M, x, y) \in L_c$, and $\pi \in \{0,1\}^*$, V_{cert} runs in time at most $g_V(k + |q|)$;

- *Completeness by a Relatively Efficient Prover*: 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 $|q| \leq k^d$,

  $$\Pr[\pi \leftarrow P_{\mathsf{cert}}(c, 1^k, q) : V_{\mathsf{cert}}(c, 1^k, q, \pi) = 1] \geq 1 - \mu(k)$$

  Furthermore, P_{cert} on input $(c, 1^k, q)$ outputs a certificate of length $\ell(k)$ in time bounded by $g_P(k + |M| + T_M(x))$.

- *Soundness*: For every $c \in N$, and every PPT P^*, there exists a negligible function μ such that for every $k \in N$,

  $$\Pr[(q, \pi) \leftarrow P^*(c, 1^k) : V_{\mathsf{cert}}(c, 1^k, q, \pi) = 1 \wedge q \notin L_c] \leq \mu(k)$$

We also consider a stronger soundness condition stipulating that soundness holds even if the attacker selects a slightly super-polynomial-size statement and specifies some slightly super-polynomial runtime.

- *Strong Soundness*: There exists some "nice" super-polynomial function[9] $T(k) \in k^{\omega(1)}$ and some "nice" super-constant function[10] $C(\cdot) \in \omega(1)$ such that for every probabilistic algorithm P^* with running-time bounded by $T(\cdot)$, there exists a negligible function μ, such that, for every $k \in N$, $c \leq C(k)$

$$\Pr[(c, q, \pi) \leftarrow P^*(1^k) : V_{\mathsf{cert}}(c, 1^k, q, \pi) = 1 \wedge q \notin L_c] \leq \mu(k)$$

We say that $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is a *statistically-sound* **P***-certificate system* (resp. *statistically sound strong* **P***-certificate system* if the soundness condition holds also against (unbounded) P^* with polynomially-bounded (resp. $T(\cdot)$-bounded) output.

Remark 1. *The reason that we do not consider a notion of computational soundness with respect to non-uniform polynomial-time attackers is that such a notion is equivalent to statistical soundness: if an accepting proof of a false statement exists, a non-uniform efficient attacker can simply get it as non-uniform advice. Nevertheless, it still makes sense to consider a notion of* $a(\cdot)$-bounded-non-uniform *soundness, where soundness holds for attacker that on input the security parameter* 1^k *additionally receive* $a(k)$ *bits of non-uniform advice. Our results regarding uniform soundness directly extend also to the regime of bounded non-uniform soundness.*

As we shall see shortly, a candidate construction of a (computationally-sound) **P**-certificate systems comes from Micali's CS-proofs [Mic00]. We also note that the assumption that statistically-sound strong **P**-certificates exists is implied by the assumption that 1) $DTIME(n^{\omega(1)}) \subseteq NP$ and 2) **NP** proofs for statements in $DTIME(t)$ can be found in time polynomial in t [BLV06]. (In essence, the assumption says that non-determinism can slightly speed-up computation, and that the non-deterministic choices can be found efficiently, where efficiency is measured in terms of the original deterministic computation.)

3.1 Time-Representation Invariant P-certificates

At first sight it may seem that since we consider only languages in **P**, the sound (resp., strongly soundness) condition of **P**-certificates is *falsifiable* [Pop63, Nao03]: we should be able to efficiently test if an attacker outputs a valid proof of an incorrect statement, since whether a statement is correct or not can be checked in deterministic polynomial time.

This intuition is somewhat misleading: recall that soundness needs to hold for *all* polynomial-time computations, where the time-bound n^c may be selected by the attacker trying to break soundness. Since there is no *a-priori* constant bound on c, the attacker may make the test (checking whether soundness was broken) run in super-polynomial-time (by selecting a large c.) The situation is even worse for the case of strongly sound **P**-certificates.

At first one may think that this issue can be easily resolved by restricting to certificate systems where the prover is asked to provide an upper-bound on the running-time of M in unary; this certainly makes the soundness condition falsifiable, but such certificates are no longer "short". We overcome this issue by allowing for a more flexible representation of (upper-bound on) the running-time of M, and restrict to *time-representation invariant* **P***-certificates*—namely proof systems where

[9] For instance, $T(n) = n^{\log \log \log n}$.

[10] For instance, $C(k) = \log \log \log n$.

whether the verifier accepts a proof of a statement x does not depend on how the time-bound is represented. For a time-invariant **P**-certificate, it suffices to define soundness in the case that the attacker specifies the running-time bound in unary; by the time-representation invariance condition, this implies soundness also for other (more efficient) representations.

Towards this, we consider an alternative variant of canonical languages in **P**: for every constant $c \in N$, let $L'_c = \{(M, x, y, 1^n) : \ M(x) = y \text{ within } n^c \text{ steps}\}$.

Definition 10. A pair of probabilistic interactive Turing machines, $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$, is a *time-representation invariant* **P***-certificate system* if there exist polynomials g_P, g_V, ℓ such that the following holds:

- *Efficient Verification:* On input $c \geq 1$, 1^k and a statement $q = (M, x, y, 1^n) \in L'_c$, and $\pi \in \{0,1\}^*$, V_{cert} runs in at most $g_V(k + |q|)$ time.
- *Time-Representation Invariant Verification:* There exists a negligible function μ such that every $c, \tilde{c}, n, \tilde{n}$, such that $n^c = \tilde{n}^{\tilde{c}}$, every $k \in N$ and every $(M, x, y) \in \{0,1\}^*$ and every certificate $\pi \in \{0,1\}^*$,
$$|\Pr[V_{\mathsf{cert}}(c, 1^k, (M, x, y, 1^n), \pi) = 1] - \Pr[V_{\mathsf{cert}}(\tilde{c}, 1^k, (M, x, y, 1^{\tilde{n}}), \pi) = 1]| \leq \mu(k)$$
- *Completeness by a Relatively Efficient Prover*: For every $c, d \in N$, there exists a negligible function μ such that for every $k \in N$ and every $q = (M, x, y, 1^n) \in L'_c$ such that $|q| \leq k^d$,
$$\Pr[\pi \leftarrow P_{\mathsf{cert}}(c, 1^k, q) : V_{\mathsf{cert}}(c, 1^k, q, \pi) = 1] \geq 1 - \mu(k)$$
Furthermore, P_{cert} on input $(c, 1^k, q)$ outputs a certificate of length at most $\ell(k)$ in time bounded by $g_P(k + |M| + n^c)$.
- *Soundness for L'_1*: For every PPT P^*, there exists a negligible function μ such that for every $k \in N$,
$$\Pr[(q, \pi) \leftarrow P^*(1^k) : V_{\mathsf{cert}}(1, 1^k, q, \pi) = 1 \wedge q \notin L'_1] \leq \mu(k)$$

We say that $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is a *strong time-representation invariant* **P***-certificate system* if there exists some "nice" $T(k) \in k^{\omega(1)}$ such that the soundness for L'_1 condition holds with respect to all probabilistic algorithms with running-time bounded by $T(\cdot)$. We say that $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is a *statistically-sound time-representation invariant* **P***-certificate system* (resp. *statistically sound strong time-representation invariant* **P***-certificate system*) if the soundness for L'_1 condition holds also against (unbounded) P^* with polynomially-bounded (resp. $T(\cdot)$-bounded) output.

Note that the soundness condition of time-representation invariant **P**-certificates is clearly falsifiable since checking whether the attacker actually outputs a statement $q \notin L'_1$ can be done in linear-time in the length of the statement, and verification of a certificate π for a statement q can be done in polynomial-time by definition.

Let us briefly outline a candidate construction of time-representation invariant **P**-certificates (where both P_{cert} and V_{cert} are deterministic).

A Candidate Construction Based on CS-proofs. Micali's CS proofs [Mic00] are obtained by first constructing a public-coin 4-round interactive argument for **NEXP** (similar to the "succinct" 4-round interactive argument for NP of [Kil95]) and then eliminating interaction through the Fiat-Shamir paradigm [FS90]: that is, the verifier's random message are generated by applying a random oracle to the prover's messages, and next the random oracle may be instantiated with a concrete family of hash function $\{h_k\}_{k\in N}$. More precisely, CS proofs are used to prove membership of the CS language L_{CS} with witness relation R_{CS} as defined in [Mic00]. A quadruple $(M, x, y, t) \in L_{\mathsf{CS}}$ iff the lengths of x and y are smaller than t and $M(x) = y$ in t steps. Roughly speaking, to prove a statement $q = (M, x, y, t)$, the prover, on input a security parameter 1^k, proceeds in two steps. In the first step, it constructs a PCP (Probabilistically Checkable Proof) [BFLS91, FGL+91] proof π' for q and computes a Merkle's hash tree [Mer89] with π' as the leaves using a hash function h_k, producing a root r. Then, in the second step, it computes a polylogarithmic number l of PCP queries, determined by the hash value $h_k(r)$; for each PCP query i, it finds the authentication path a_i that reveals the corresponding PCP answer b_i. Finally, the prover sends a CS proof $\pi = t\|r\|b_i\|a_i\| \cdots \|b_l\|a_l$. The verifier, on input a statement x and such a proof π, checks whether all the authentication paths are accepting w.r.t. r, recomputes the PCP queries using $h_k(r)$ and checks whether all the PCP answers are accepting.

In our language L'_c, recall that a statement is of form $q = (M, x, y, 1^n)$. The prover and the verifier on input c, 1^k and q can thus recover a time bound t by computing n^c and then recover the corresponding CS language instance (M, x, y, t), and next simply runs the prover and verifier algorithms of CS-proofs. By construction it follows that the above construction satisfies prover's relative efficiency and completeness. Additionally, since the verification procedure only depends on the time bound $t = n^c$, and not on the values of n and c, the verification procedure also has the time-representation invariance property.

Finally, in our eyes, assuming that the above construction satisfies the soundness condition of time-representation invariant $\mathbf{P}$-certificates is a reasonable and "well-behaved" complexity theoretic assumption: on a qualitatively level, the assumption is not very different from the assumption that e.g., the Full Domain Hash [BR93] or Schnorr [Sch91] signature schemes are existentially unforgeable: 1) whether an attacker succeeds can be efficiently checked, 2) no attacks are currently known, and 3) the "design-principles" underlying the constructions rely on similar intuitions (e.g., that instantiating random-oracles with hash functions in "natural" schemes lead to secure protocol).

From Time-Representation Invariant P-certificates to P-certificates As we now show, time-representation invariant $\mathbf{P}$-certificates imply $\mathbf{P}$-certificates.

Theorem 1. *Assume the existence of a time-representation invariant* $\mathbf{P}$*-certificate system (resp. a strong time-representation invariant* $\mathbf{P}$*-certificate system)* $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$*. Then, there exists a* $\mathbf{P}$*-certificate system (resp. a strong* $\mathbf{P}$*-certificate system)* $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$*. Furthermore if* $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$ *is statistically sound (resp. statistically strong sound), then* $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ *is so as well.*

Proof. Let $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$ be a time-representation invariant $\mathbf{P}$-certificate system. Consider a $\mathbf{P}$-certificate system $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ where P_{cert} and V_{cert} simply runs P'_{cert} and V'_{cert} respectively with n fixed to the length of the input x. More precisely, P_{cert} on input c, 1^k and a statement $q = (M, x, y) \in L_c$, lets $q' = (M, x, y, 1^{|x|}) \in L'_c$, runs $P'_{\mathsf{cert}}(c, 1^k, q')$ and outputs whatever P'_{cert} outputs.; V_{cert} on input $(c, 1^k, q, \pi)$ computes q' in exactly the same way, runs $V'_{\mathsf{cert}}(c, 1^k, q', \pi)$ and outputs the verdict of V'_{cert}. It follows from the relative prover efficiency and completeness properties of $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$ that $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ also satisfies relative prover efficiency and completeness. Let us turn to soundness. We only prove the case of strong soundness (assuming that $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is strongly sound), all the other cases follow analogously.

Assume for contradiction that for every $T(k) \in k^{\omega(1)}$ and $C(k) \in \omega(1)$, there exists a $T(k)$-time cheating prover A, and a polynomial p such that for infinitely many $k \in N$ and $c_k \leq C(k)$, it holds that the probability that $A(1^k)$ outputs c_k, a false statement $q = (M, x, y) \notin L_{c_k}$ and a certificate π for $q \in L_{c_k}$ that is accepted by V_{cert} (that is, $V_{\mathsf{cert}}(c_k, 1^k, q, \pi) = 1$) is at least $1/p(k)$.

Fix some arbitrary function $T'(k) \in k^{\omega(1)}$. Let $T(k) \in k^{\omega(1)}$ and $C(k) \in \omega(1)$ be two functions such that $T(k)^{C(k)} \leq T'(k)$. By our assumption, there exists a cheating prover A that violates the strong soundness property of $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ w.r.t. the functions $T(k)$ and $C(k)$ with some polynomial probability $1/p(k)$. Using A, we construct another cheating prover A' that violates the strong soundness for L'_1 of $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$ w.r.t. function $T'(k)$ with the same probability $1/p(k)$. Machine A' on input 1^k simply runs $A(1^k)$ to obtain c_k, $q = (M, x, y)$ and π; it then sets $n = |x|^{c_k}$ and outputs $q' = (M, x, y, 1^n)$ and π. Clearly, A' runs in time $T(k)^{C(k)} \leq T'(k)$. By construction of V_{cert}, the probability that $V_{\mathsf{cert}}(c_k, 1^k, q, \pi) = 1$ is the same as the probability that $V'_{\mathsf{cert}}(c_k, 1^k, \tilde{q}, \pi) = 1$, where $\tilde{q} = (M, x, y, 1^{|x|})$. Furthermore, by the time-representation-invariance of V'_{cert}, the probability that $V'_{\mathsf{cert}}(c_k, 1^k, \tilde{q}, \pi) = 1$ is negligibly close to the probability that $V'_{\mathsf{cert}}(1, 1^k, q', \pi) = 1$. It follows that A' (whose running-time is bounded by $T'(k)$) outputs accepting proofs of false statements with probability negligibly close to $\frac{1}{p(k)}$ for infinitely many $k \in N$. Since the above holds for any function $T'(k)$, we have that $(P'_{\mathsf{cert}}, V'_{\mathsf{cert}})$ is not strongly sound for L'_1, which is a contradiction. □

4 Constant-round Concurrent $\mathcal{ZK}$

In this section, we present our construction of a constant-round concurrent $\mathcal{ZK}$ protocol. To simplify the exposition (and following the description in the introduction), as a warm-up, we first present a protocol that only uses one level of **P**-certificates and thus only handles a bounded number, $O(m(n))$, of concurrent executions; we refer to this protocol as "Protocol 1". We then generalize Protocol 1 and describe a protocol that uses k levels of certificates and can handle $O(n^k)$ concurrent executions; we refer to this protocol as "Protocol k". By setting k to be super-constant, say, $k = \log n$, we obtain a (fully) concurrent $\mathcal{ZK}$ protocol.

4.1 Protocol 1

We proceed to describe Protocol 1, (P_1, V_1), which we prove is m-bounded concurrent zero-knowledge. The protocol relies on the following primitives:

- A commitment scheme com: for simplicity of presentation, we assume that com is a non-interactive commitment scheme, but the protocol can be modified in a straight-forward way to work for any two-message commitment scheme (as in [Nao91]).
- A strong **P**-certificate system $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ 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: for, simplicity of exposition, we assume that both P_{cert} and V_{cert} are *deterministic*. We discuss in Section 4.3 how to modify the protocol to also handle randomized **P**-certificate systems.
- 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 [BG02], this assumption can be weakened to just collision resistance against polynomial-size circuits by modifying the protocol to use a "good" error-correcting code ECC (i.e., with constant distance and with polynomial-time encoding and decoding), and replace commitments $\mathsf{com}(h(\cdot))$ with $\mathsf{com}(h(\mathsf{ECC}(\cdot)))$.

Let us now turn to specifying the protocol (P_1, V_1). 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 **P**-certificates, we always consider the language L_D.

The prover P_1 and the verifier V_1, on common input 1^n and x and private input a witness w to P_1, proceed as follow:

Phase 1: P_1 and V_1 exchanges the following three messages.

1. V_1 chooses a randomly sampled hash function $h \leftarrow \mathcal{H}_n$.
2. P_1 sends a commitment to 0^n using com.
3. V_1 replies with a random "challenge" r of length $3mn$.

Phase 2: P_1 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, such that

1. **Commitment Consistency:** $c = \mathsf{com}(h(\tilde{S}_1); \rho)$,
2. **Input Certification:** $|\sigma| \leq 2mn$,
3. **Prediction Correctness:** π certifies that $\tilde{S}_1(1^n, j, s, \sigma) = r$.

A formal description of the protocol can be found in Figure 4 and 5.

Protocol 1

Common Input: A security parameter 1^n in unary and an instance x of a language $L \in \mathbf{NP}$ with witness relation $\mathbf{R}_L$.

Parameters: $m = m(n)$ is an upper bound on the number of concurrent sessions. $\Gamma = \Gamma(n)$ and $D = D(n)$ are respectively upper bounds on the size of the committed program and the time bound.

Phase 1:

$V_1 \to P_1$: Send $h \leftarrow \mathcal{H}_n$.

$P_1 \to V_1$: Send $c = \mathsf{com}(0^n; \rho)$.

$V_1 \to P_1$: Send $r \leftarrow \{0,1\}^{3mn}$.

Phase 2:

$P_1 \Leftrightarrow V_1$: A WIUA $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}} \rangle$ proving the OR of the following statements:

1. $\exists\, w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $\mathbf{R}_L(x, w) = 1$.
2. $\exists\, \langle \tilde{S}_1, j, s, \pi, \sigma, \rho \rangle$ s.t. $\mathbf{R}_S(\langle h, c, r \rangle, \langle \tilde{S}_1, j, s, \pi, \sigma, \rho \rangle) = 1$.

Figure 4: A public-coin non-black-box bounded concurrent zero-knowledge protocol.

Our Simulator. As explained in the introduction, the goal of our simulator is to try to "commit to its own code" in a way that prevents a blow-up in the running-time. Note that in our 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

Instance: A triplet $\langle h, c, r\rangle \in \mathcal{H}_n \times \{0,1\}^{\mathrm{poly}(n)} \times \{0,1\}^{3mn}$.

Witness: $\langle \tilde{S}_1, j, s, \pi, \sigma, \rho\rangle$: A program $\tilde{S} \in \{0,1\}^{\Gamma}$, an integer $j \in [m]$, a seed $s \in \{0,1\}^n$, a **P**-certificate $\pi \in \{0,1\}^n$, a string $\sigma \in \{0,1\}^{\Gamma}$, a randomness $\rho \in \{0,1\}^n$.

Relation: $\mathbf{R}_S(\langle h, c, r\rangle, \langle \tilde{S}_1, j, s, \pi, \sigma, \rho\rangle) = 1$ if and only if:

1. **Commitment Consistency:** $c = \mathsf{com}(h(\tilde{S}_1); \rho)$,
2. **Input Certification:** $|\sigma| \leq 2mn$,
3. **Prediction Correctness:** $V_{\mathsf{cert}}(D, 1^n, (\tilde{S}_1, (1^n, j, s, \sigma), r), \pi) = 1$ (i.e., π certifies that $\tilde{S}_1(1^n, j, s, \sigma) = r$).

Figure 5: $\mathbf{R}_S$, a relation that Protocol 1 uses in WIUA of Phase 2.

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. Roughly speaking, 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"; these "left" messages will be certificates provided by S_2. See Figure 1 in the introduction for an illustration of the communication patterns between S_1, S_2 and V^*.

Let us turn to a formal description of the S_1 and S_2. To simplifiy 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 receiving some additional "external" messages 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).

Let us now specify how S_1 generates prover messages in its "right" interaction with V^*. $S_1(1^n, x, M, s, \ell)$ acts as follows:

- 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 of 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.

- 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 program $\tilde{S}_1(1^n, j, s', \tau) = \mathit{wrap}(M(1^n, x, M, s', j), V^*, \tau, j)$, where $\mathit{wrap}(A, B, \tau, j)$ is the program that lets A communicate with B for j rounds, while allowing A to receive τ as external messages, and finally outputting B's message in the j'th communication round. (That is, $\tilde{S}_1(1^n, j, s', \tau)$ emulates j rounds of an execution between S_1 and V^* where S_1 expands out the seed s' into j blocks of randomness and additionally receives τ as external messages.)

- Upon receiving a challenge r_i in session i during the j'th communication round, S_1 needs to provide a WIUA. To do so, it checks whether it was received an external message (j, π_j), and if so, it uses the certificate π_j to complete the WIUA (and otherwise halts). More precisely, S_1 provides an honest WIUA that c_i is a commitment to $\tilde{S}_1$ and that π_j certifies that $\tilde{S}_1(1^n, j, s_j, \tau) = r_i$ where τ is list of external messages received by S_1 so far. (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.)

$S_2(1^n, x, M, s, \ell)$ internally emulates ℓ messages of an execution between $S_1(1^n, x, M, s, \ell)$ and V^*. In *each* communication round j, after V^* generates a verifier message m_j, S_2 generates a certificate π_j (using P_{cert}) that $\tilde{S}_1(1^n, j, s_j, \sigma) = m_j$, where σ is the list of external messages received by S_1 so far, and feeds (j, π_j) to S_1. Finally, S_2 outputs its view (which in particular, contains the view of V^*) at the end of the execution.

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.

Running-time of S. Let us first argue that S_1 runs in polynomial time. Clearly it only takes S_1 polynomial-time to generate the commitments in Phase 1 (since V^* has a polynomial-length description, and thus also the code of S_1). During the WIUA in Phase 2, the length of the witness used by the simulator is polynomial in length of the description of $\tilde{S}_1$ and the length of the certificate π used by S_1; both are of polynomial length. Since the $\mathbf{P}$-certificates verification time is polynomial in the length of the statement proved, it follows that the relation being proved in the WIUA has a time complexity that is upper bounded by a fixed polynomial in the length of V^*. By the relative prover efficiency condition of the WIUA, each such proof only requires some fixed polynomial-time, and thus 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 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

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

Consider a hybrid experiment $\mathsf{Real}'_{V^*}(n, x, z)$ that proceeds just as the real experiment except that all phase 1 commitments are generated by committing to the code of $\tilde{S}_1$ (as done by S). We also denote by $\mathsf{Real}'_{V^*}(n, x, z)$ the view of the verifier V^* in the hybrid. It follows by a simple hybrid argument that there exists a polynomial p' such that the view of V^* in the hybrid Real' and in simulation by S can be distinguished by D with probability $\frac{1}{p'(n)}$ for infinitely many n. That is, 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

$$\left|\Pr[D(\mathsf{Real}'_{V^*}(n,x,w,z))) = 1] - \Pr[D(S(1^n,x,z)) = 1]\right| \geq \frac{1}{p'(n)} \qquad (1)$$

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 simulated by S (and more specifically by S_1) using *true* randomness q'_j being uniformly distributed in $\{0,1\}^n$; additionally, to complete all WIUA that *begin at or after* communication round j, S_1 uses the true witness w instead of the "fake" witness used by S_1. (Note that once we start using real randomness is some session i, it is no longer clear whether simulation of "later" sessions can be completed. To deal with this issue, we thus also switch all WIUA that begin at or after round j to use a real witness; if some WIUA already began at some communication round $j' < j$, then the simulation of this WIUA can still be completed.)

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, but where true randomness is used in communication round $j+1$ (but still using the fake witness). 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 used by S_1 to provide WIUA's). Indistinguishability of $\tilde{H}_j$ and H_j follows directly by the witness indistinguishability property of the WIUA. It thus leads to a contradiction and completes the proof of the indistinguishability of the simulation.

4.2 Protocol k

We move on to describe our actual concurrent $\mathcal{ZK}$ protocol: Protocol k, (P_k, V_k). We refer the reader to the introduction for the ideas underlying this protocol.

As with Protocol 1, Protocol k proceeds in two phases. In Phase 1, the prover P_k and the verifier V_k proceeds exactly as in Protocol 1 but the length of the "challenge" r is modified to be $3kn^2$. Next, Phase 2 is modified as follows:

Phase 2: P_k gives a WIUA argument of the statement that either $x \in L$ OR there exists $\tilde{S}_1, \ldots, \tilde{S}_k \in \{0,1\}^{\Gamma(n)}$, $0 < j < n^k$, $s^1, \ldots, s^k \in \{0,1\}^n$, $\pi^1, \ldots, \pi^k \in \{0,1\}^n$, $\sigma^1, \ldots, \sigma^k \in \{0,1\}^{\Gamma(n)}$, $\lambda^1, \ldots, \lambda^k \in \{0,1\}^{\Gamma(n)}$, ρ, such that

1. **Commitment Consistency:** $c = \mathsf{com}(h(\tilde{S}_1, \ldots, \tilde{S}_k); \rho)$,
2. **Input Certification:**
   (a) $|\vec{\sigma}| \leq 2kn^2$; and
   (b) Let l^* be the largest l such that $j \geq n^{l-1}$. Then $\lambda^{\geq l^*} = \mathsf{null}$ and for $2 \leq l \leq l^*$, π^l certifies that $\tilde{S}_l(1^n, \lfloor j \rfloor_{n^{l-1}}, s^l, ([\lambda^{\geq l}]_{\lfloor j \rfloor_{n^{l-1}}}, \sigma^{\geq l})) = \lambda^{l-1}$.
3. **Prediction Correctness:** π^1 certifies that $\tilde{S}_1(1^n, j, s^1, ([\lambda^{\geq 1}]_j, \sigma^{\geq 1}))) = r$

where $\lfloor j \rfloor_x \triangleq j - (j \mod x)$, and the bracket operator $[\cdot]_j$ is defined as follows: The input is expected to be a set of triples of the form $(j', l', \pi^{l'}_{j'})$, and the output is a subset of these obtained by removing elements with $j' \geq j$; that is, we are "filtering out" all messages that were generated in communication round j or later. Roughly speaking, the bracket operator is used to eliminate "unnecessary" inputs to the program. We require this to be able to reuse **P**-certificates; we provide a more detailed explanation of why this is needed in Remark 2, after having formalized the simulator.

Using the notation from the introduction, the messages $\vec{\lambda}$ are "certified" certificates (each component of $\vec{\lambda}$ may of an unbounded polynomial length), and the messages $\vec{\sigma}$ are "dangling" certificates (each component of $\vec{\sigma}$, however, is "short" by the input certification condition).

A formal description of Protocol k can be found in Figure 6 and 7.

We will be analyzing (P_k, V_k) when $k = \log n$ (but the analysis works as long as k is a "nice" super-constant, but polynomially-bounded, function). It is easy to check that the protocol is complete. Furthermore, since the honest prover P_k, on private input a valid witness w of the statement x, always succeeds in the Phase 2 by proving that $x \in L$, by the prover and verifier efficiency conditions of WIUA, both the honest prover P_k and verifier V_k run in some fixed polynomial time. Furthermore note that the communication complexity of the protocol depends only on the security parameter 1^n but not the length of the statement x; thus the protocol is "succinct".

We turn to showing that (P_k, V_k) is sound and concurrent $\mathcal{ZK}$ when $k = \log n$.

PROTOCOL k

Common Input: A security parameter 1^n and an instance x of a language $L \in \mathbf{NP}$ with witness relation $\mathbf{R}_L$.

Parameters: $m = m(n)$ is an upper bound on the number of concurrent sessions. $\Gamma = \Gamma(n)$ and $D = D(n)$ are respectively upper bounds on the size of the committed program and the time bound.

Phase 1:

$V_k \to P_k$: Send $h \leftarrow \mathcal{H}_n$.

$P_k \to V_k$: Send $c = \mathsf{com}(0^n; \rho)$.

$V_k \to P_k$: Send $r \leftarrow \{0,1\}^{3kn^2}$.

Phase 2:

$P_k \Leftrightarrow V_k$: A WIUA $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}} \rangle$ proving the OR of the following statements:

1. $\exists\ w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $\mathbf{R}_L(x, w) = 1$.
2. $\exists\ \langle \vec{\vec{S}}, j, \vec{s}, \vec{\pi}, \vec{\sigma}, \vec{\lambda}, \rho \rangle$ s.t. $\mathbf{R}_S(\langle h, c, r \rangle, \langle \vec{\vec{S}}, j, \vec{s}, \vec{\pi}, \vec{\sigma}, \vec{\lambda}, \rho \rangle) = 1$.

Figure 6: A public-coin non-black-box concurrent zero-knowledge protocol.

4.2.1 Soundness of Protocol k

Lemma 1. *Under the above-mentioned cryptographic assumptions, (P_k, V_k) is uniformly sound. Additionally, if $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is a statistically strong **P**-certificate system, then (P_k, V_k) is non-uniformly sound.*

Proof. We prove this lemma w.r.t. uniform soundness assuming $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ is a strong **P**-certificate; the non-uniform part of the lemma follows in identically the same way.

Instance: A triplet $\langle h, c, r\rangle \in \mathcal{H}_n \times \{0,1\}^{\mathrm{poly}(n)} \times \{0,1\}^{3kn^2}$.

Witness: $\langle \vec{\tilde{S}}, j, \vec{s}, \vec{\pi}, \vec{\sigma}, \vec{\lambda}, \rho\rangle$: A sequence of programs $\vec{\tilde{S}} = (\tilde{S}_1, \ldots, \tilde{S}_k) \in \{0,1\}^{k\cdot\Gamma}$, an integer $j \in [n^k]$, a sequence of seeds $\vec{s} = (s^1, \ldots, s^k) \in \{0,1\}^{k\cdot n}$, a sequence of **P**-certificates $\vec{\pi} = (\pi^1, \ldots, \pi^k) \in \{0,1\}^{k\cdot n}$, a sequence $\vec{\sigma} = (\sigma^1, \ldots, \sigma^k) \in \{0,1\}^{k\cdot\Gamma}$, a sequence $\vec{\lambda} = (\lambda_1, \ldots, \lambda_k) \in \{0,1\}^{k\cdot\Gamma}$, a randomness $\rho \in \{0,1\}^n$.

Relation: $\mathbf{R}_S(\langle h, c, r\rangle, \langle \vec{\tilde{S}}, j, \vec{s}, \vec{\pi}, \vec{\sigma}, \vec{\lambda}, \rho\rangle) = 1$ if and only if:

1. **Commitment Consistency:** $c = \mathsf{com}(h(\vec{\tilde{S}}); \rho)$,
2. **Input Certification:**
   (a) $|\vec{\sigma}| \leq 2kn^2$, and
   (b) Let l^* be the largest l such that $j \geq n^{l-1}$. $\lambda^{\geq l^*} = \mathsf{null}$ and for $2 \leq l \leq l^*$, $V_{\mathsf{cert}}(D, 1^n, (\tilde{S}_l, (1^n, \lfloor j\rfloor_{n^{l-1}}, s^l, ([\lambda^{\geq l}]_{\lfloor j\rfloor_{n^{l-1}}}, \sigma^{\geq l})), \lambda^{l-1}), \pi^l) = 1$ (i.e., π^l certifies that $\tilde{S}_l(1^n, \lfloor j\rfloor_{n^{l-1}}, s^l, ([\lambda^{\geq l}]_{\lfloor j\rfloor_{n^{l-1}}}, \sigma^{\geq l})) = \lambda^{l-1}$).
3. **Prediction Correctness:** $V_{\mathsf{cert}}(D, 1^n, (\tilde{S}_1, (1^n, j, s^1, ([\lambda^{\geq 1}]_j, \sigma^{\geq 1})), r), \pi^1) = 1$ (i.e., π certifies that $\tilde{S}_1(1^n, j, s^1, ([\lambda^{\geq 1}]_j, \sigma^{\geq 1})) = r$).

where $\lfloor j\rfloor_x \triangleq j - (j \mod x)$, and the operator $[\cdot]_j$ is defined as follows: The input is expected to be a set of triples of the form $(j', l', \pi^{l'}_{j'})$, and the output is a subset of these obtained by removing elements with $j' \geq j$.

Figure 7: $\mathbf{R}_S$, a relation that Protocol k uses in WIUA of Phase 2.

Assume for contradiction that there is a probabilistic polynomial time cheating prover P^* and a polynomial p, such that for infinitely many $n \in \mathbb{N}$, with probability $1/p(n)$, P^* selects a false statement $x \in \{0,1\}^{\mathrm{poly}(n)} \setminus L$ and convinces V_k of the membership of x in L.

Fix one such n. Let $P^*_{u,h,r}$ be the "residual" deterministic WIUA prover resulting from fixing P^*'s randomness to u and feeding it the messages h and r. Let E be the "global" proof-of-knowledge extractor of the WIUA. Note that E runs in time $\mathrm{poly}(\Gamma(n))$. Let E_s denote E with randomness fixed to s. Now, consider the following experiment Exp:

- Sample a tuple (u, h, r, s) uniformly at random.
- Let $(x, c) \leftarrow P^*_{u,h,r}$ and $w' \leftarrow E_s^{P^*_{u,h,r}}$, where x is the statement selcted by $P^*_{u,h,r}$, c is the commitment generated by $P^*_{u,h,r}$, and w' is the witness extracted by $E_s^{P^*_{u,h,r}}$.

Let BAD denote the event that $E_s^{P^*_{u,h,r}}$ extracts a valid "fake" witness $w' = (\vec{\tilde{S}}, j', \vec{s}', \vec{\pi}', \vec{\sigma}', \vec{\lambda}', \rho') \in \mathbf{R}_S(h, c, r)$ in the above experiment.

Let us first argue that by our assumption (that P^* breaks soundness), BAD happens with non-negligible probability: By an averaging argument, with probability at least $1/2p(n)$ over (u, h, r), the statement x selected by $P^*_{u,h,r}$ is not a member of L and yet $P^*_{u,h,r}$ convinces the WIUA verifier with probability $1/2p(n)$. For each such a tuple (u, h, r), by the "global" proof-of-knowledge property of WIUA, $E_s^{P^*_{u,h,r}}$ extracts a valid "fake" witness $w' \in \mathbf{R}_S(h, c, r)$ with some non-negligible probability $1/q(n)$ (over the randomness s). It follows that BAD happens with non-negligible probability.

We now show that under our cryptographic assumptions, BAD can only happen with negligible probability, which is a contradiction.

First, note that by the soundness of $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ with parameters $T(\cdot)$ and $C(\cdot)$, and the fact that $T(n) = \Gamma(n)^{\omega(1)}$ and $D(n) \leq C(n)$, we have that except with negligible probability over the choice of (u, h, r, s), whenever the **P**-certificates $\vec{pi}'$ that $E_s^{P^*_{u,h,r}}$ extracts out are convincing, their corresponding statements are true; otherwise, we can construct a uniform poly$(\Gamma(n))$-time adversary that samples u, h, r, s uniformly at random, runs $E_s^{P^*_{u,h,r}}$, and outputs a random certificate from w'. Additionally, by the binding property of com and the collision-resistant property of $\mathcal{H}_n$ it follows that with overwhelming probability over (u, h), there exists a vector of machines $\vec{\tilde{S}}^*$ such that except with negligible probability over the choice of r, s, it holds that if $E_s^{P^*_{u,h,r}}$ outputs a valid $w' \in \mathbf{R}_S(h, c, r)$, then the machines $\vec{\tilde{S}}$ in w' equals $\vec{\tilde{S}}^*$.[11] By a union bound it follows that with overwhelming probability over (u, h), there exists a vector of machines $\vec{\tilde{S}}^*$ such that except with negligible probability over the choice of r, s, the following holds: a) if $E_s^{P^*_{u,h,r}}$ outputs a valid $w' \in \mathbf{R}_S(h, c, r)$, then the machines $\vec{\tilde{S}}$ in w' equals $\vec{\tilde{S}}^*$, and b) all accepting certificates $\vec{\pi}'$ prove true statements. Let us refer to such pairs (u, h) as good.

For any valid "fake" witness $w' = (\vec{\tilde{S}}, j', \vec{s}', \vec{\pi}', \vec{\sigma}', \vec{\lambda}', \rho') \in \mathbf{R}_S(h, c, r)$ define a machine $M_{w'}$ (using $\vec{\tilde{S}}$ in w') that given the input $(j', \vec{s}', \vec{\sigma}')$ of length smaller than $2kn^2$, outputs r:

Machine $M_{w'}$: $M_{w'}(1^n, j, \vec{s}, \vec{\sigma})$ lets l^* be the largest l such that $j > n^{l-1}$. $M_{w'}$ next runs the machines $\tilde{S}_{l^*}, \tilde{S}_{l^*-1}, \ldots, S_1$ in sequence as follows: $\tilde{S}_{l^*}$ is run on input $1^n, j^{l^*}$, s^{l^*} and σ^{l^*}; let λ^{l^*-1} denote its output. Next for each $l \leq l^*$, $\tilde{S}_l$ is given 1^n, j^l, $s^l, \sigma^{\geq l}$ and $[\lambda^{\geq l}]_{j,l}$ where $\lambda^{\geq l}$ are the outputs of the executions of $\tilde{S}_{l+1}, \ldots, \tilde{S}_{l^*}$. Finally, M outputs the string r returned by $\tilde{S}_1$.

Note that by definition, if all the **P**-certificates in w' prove true statements, then $M_{w'}$ given the input $(j', \vec{s}', \vec{\sigma}')$ indeed outputs r. However, for any machine M, since the input to the machine M is of length $2kn^2$, it follows by a counting argument that only for a negligible fraction of length $3kn^2$ strings r, there exists some input that makes M output r. Thus, whenever (u, h) is good (which happens with overwhelming probability), except with negligible probability (over the choice of r, s) **BAD** cannot happen; it follows that **BAD** can only happen with negligible probability, which is a contradiction.

$\square$

4.2.2 Concurrent $\mathcal{ZK}$ of Protocol k

The simulator S for Protocol k will define $k+1$ "helper" simulators $S_1, \ldots, S_{k+1}$. Before providing the formal definition of $S_1, \ldots, S_{k+1}$, let us first describe the interaction among them.

Recall that in the simulation of Protocol 1, S_1 is an interactive machine that communicates with a concurrent verifier V^*, on the "right", while expecting to receive a **P**-certificates (j, π_j) from S_2, on the "left", for *every* communication round j in the right interaction with V^*; S_1 then makes use of these certificates to complete the right interaction with V^* (and more specifically, to complete the WIUAs it is supposed to provide V^*). In the simulation of Protocol k, S_1 still communicates with V^* on the "right", but now additionally expects to receive **P**-certificates from

[11] Note that for this to hold, we here rely on the fact that binding of com and collision-resistancy of $\mathcal{H}_n$ hold also for circuits of size poly$(\Gamma(n))$; however, as mentioned, by slightly modifying the protocol as in [BG02], this assumption can be weakened to just collision resistance against polynomial-size circuits.

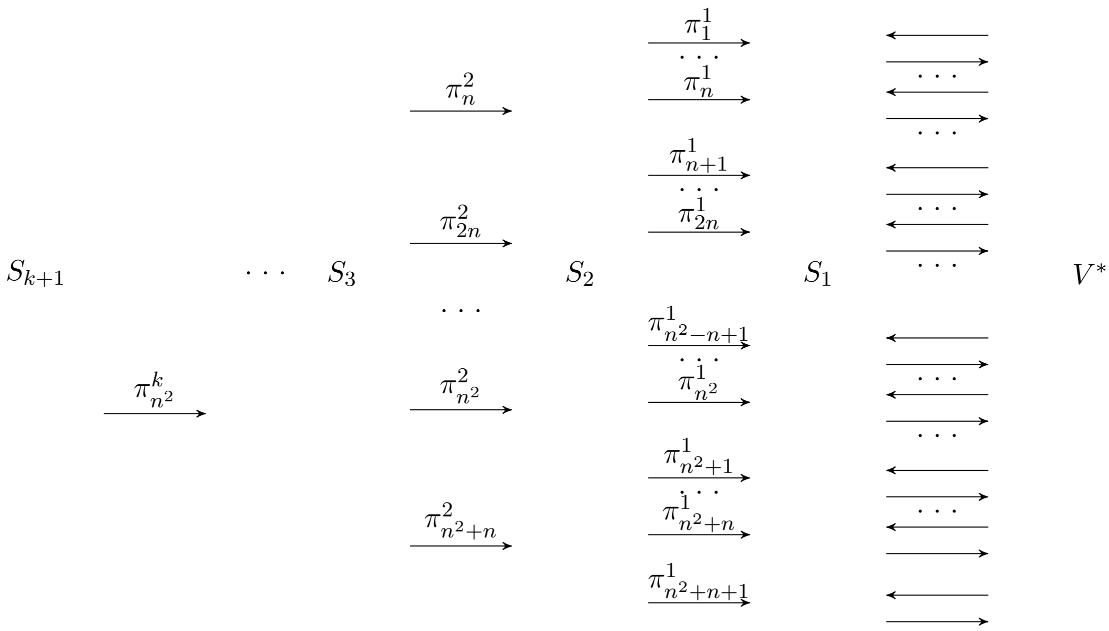

Figure 8: Simulation of protocol (P_k, V_k) for $k = 3$.

all of $S_2, \dots, S_{k+1}$ on the "left". In more detail, recall that a communication round in the "right" interaction refers to a verifier message (sent by V^*) followed by a prover message (sent by S_1). Now, in each communication round j in the right interaction, upon receiving a message from the verifier V^*, S_1 also expects to receive $(j, 1, \pi_j^1)$ from S_2, and furthermore, for every $2 \leq l \leq k$, if $j \mod n^{l-1} = 0$, then S_1 additionally expects to receive (j, l, π_j^l) from S_{l+1}. In other words, S_1 expects to receive a "level-l" certificate (of the form $(j = a \cdot n^{l-1}, l, \pi_j^l)$ for some a) from S_{l+1} every n^{l-1} communication rounds. Roughly speaking, each such "level-l" certificatate, certifies that all "level-$(l-1)$" certificates up to round j were actually generated by S_l; and those "level-$(l-1)$" certificates certify that S_{l-1} actually generated the "level-$(l-2)$" certificates up until round j, etc. See Figure 8 for an illustration of the communication pattern between $V^*, S_1, \dots, S_{k+1}$.

For every $2 \leq l \leq k$, for S_l to be able to generate its level $(l-1)$-certificates, S_l internally emulates the interaction among $S_{l-1}, \dots, S_1, V^*$, but additionally needs to receive all level-l' certificates, where $l' \geq l$; thus each machine S_l produces level-$l-1$ certificates on the "right", while receiving level-l, level-$(l+1)$, ... level-k certificates from respectively S_{l+1}, $S_{l+2}, \dots S_{k+1}$, on the "left". See Figure 9 for an illustration of S_l.

We now define S_1. As before, on a high-level, $S_1(1^n, x, \vec{M}, s, \ell)$, acts as a prover in a "right" interaction, communicating with a concurrent verifier V^*, while receiving some additional "external" messages on the "left". (The input x is the statement to be proved, the input $\vec{M}$ will later be instantiated with the codes of $S_1, \dots S_k$, 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.)

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

- Upon invocation, S_1 generates its "random-tape" by expanding the seed s; more specifically, let $(s_\ell, s_{\ell-1}, \dots s_1)$, $(q_\ell, q_{\ell-1}, \dots, q_1)$ be the output of $g(s, \ell)$. Again, we assume without loss of generality that S_1 only needs n bits of randomness of generate any prover message; in order

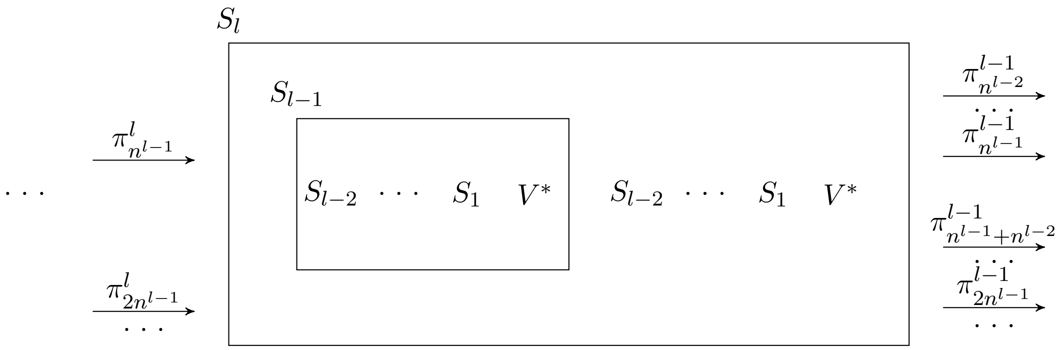

Figure 9: Simulator S_l.

to generate its j'th prover message, it uses q_j as randomness.

- Upon receiving a hash function h_i for session i in communication round j, S_1 provides a commitment c_i to the hash of the programs $\tilde{S}_1, \ldots, \tilde{S}_k$ defined as follows.
  - $\tilde{S}_1(1^n, j, s', \sigma) = \mathit{wrap}(M_1(1^n, x, \vec{M}, s', j), V^*, \sigma, j)$.
  - For $2 \le l \le k$, $\tilde{S}_l(1^n, j, s', \sigma) = \mathit{wrap}'(M_l(1^n, x, \vec{M}, s', j), \sigma, j)$ where $\mathit{wrap}'(A, \sigma, j)$ is the program that executes A for j "communication rounds," while allowing A to receive σ as external messages "on the left", and finally outputs the set of messages generated by A "on the right"—recall that M_l will be instantiated by S_l, who emulates the interaction among $S_{l-1}, \ldots, S_1, V^*$, receives level-$l'$ certificates for $l' \ge l$ externally "on the left", and generates level-$(l-1)$ certificates on the "right"; "communication rounds" here still refer to the communication rounds of S_1 and V^*. (wrap' simply returns $\perp$ whenever A does not have the specified structure.)

- Upon receiving a challenge r_i in session i during the jth communication round, S_1 needs to provide a WIUA. To do so, S_1 collects the witness as follows.
  - Let l^* be the largest l such that $j \ge n^{l-1}$.
  - For $1 \le l \le l^*$, set $s^l = s_{\lfloor j \rfloor_{n^{l-1}}}$ (i.e., the seed corresponding to communication rounds $\lfloor j \rfloor_{n^{l-1}}$; recall that $\lfloor j \rfloor_x \triangleq j - (j \mod x)$).
  - For $1 \le l \le l^*$, recall that S_1 expects to have received $a_l = \lfloor j \rfloor_{n^{l-1}} / n^{l-1}$ messages from S_{l+1} of the form $(a \cdot n^{l-1}, l, \pi^l_{a \cdot n^{l-1}})$ for $a \in [a_l]$.
    - ∗ Let π^l be the **P**-certificate in the last message received from S_{l+1}; by construction, this message was received in round $\lfloor j \rfloor_{n^{l-1}}$ and thus we have $\pi^l = \pi^l_{\lfloor j \rfloor_{n^{l-1}}}$.
    - ∗ Let λ^l be the messages received from S_{l+1} up until and including round $\lfloor j \rfloor_{n^l}$; by construction, since S_{l+1} generates a message every n^{l-1} communication rounds, λ^l contains a total of $\lfloor j \rfloor_{n^l} / n^{l-1}$ messages.
    - ∗ Let σ^l be the messages generated by S_{l+1} after round $\lfloor j \rfloor_{n^l}$ but before round $\lfloor j \rfloor_{n^{l-1}}$ (thus, we exclude the last message π^l and the messages included in λ^l); since there are at most n^l communication rounds after round $\lfloor j \rfloor_{n^l}$ and before round $\lfloor j \rfloor_{n^{l-1}}$, and (again) S_{l+1} generates a message every n^{l-1} rounds, σ^l contains at most n messages; each such message is of length $n + O(\log n) < 2n$.
  - For $l^* < l \le k$, let $\lambda_l = \mathsf{null}$. (Note that also $\lambda_{l^*} = \mathsf{null}$ since $\lfloor j \rfloor_{n^{l^*}} = 0$.)

- Finally, let ρ and $\vec{\tilde{S}}$ be the randomness and machines, respectively, used to generate the commitment c_i in the i^{th} session.

If S_1 fails to find a valid witness, S_1 simply halts. Otherwise, S_1 uses the above witness to provide an honest WIUA to V^* that

1. (Commitment consistency:) $c_i = \mathsf{com}(h_i(\tilde{S}_1, \ldots, \tilde{S}_k); \rho)$,
2. (Input certification:) $|\vec{\sigma}| \leq 2kn^2$, $\lambda^{\geq l^*} = \mathsf{null}$ and for $2 \leq l \leq l^*$, π^l certifies that $\tilde{S}_l(1^n, \lfloor j \rfloor_{n^{l-1}}, s^l, ([\lambda^{\geq l}]_{\lfloor j \rfloor_{n^{l-1}}}, \sigma^{\geq l})) = \lambda^{l-1}$,
3. (Prediction correctness:) π^1 certifies that $\tilde{S}_1(1^n, j, s^1, ([\lambda^{\geq 1}]_j, \sigma^{\geq 1})) = r_i$

Remark 2. *Above, for every $1 \leq l \leq l^*$, S_1 uses the $\mathbf{P}$-certificates π^l to certify that the execution of $\tilde{S}_l$ up until communication round $\lfloor j \rfloor_{n^{l-1}}$ when providing $\tilde{S}_l$ with the "certified" inputs $[\lambda^{\geq l}]_{\lfloor j \rfloor_{n^{l-1}}}$ and "dangling" inputs $\sigma^{\geq l}$. The bracket operator is used to ensure that the inputs given to $\tilde{S}_l$ are identically the same as were given to it when generating the $\mathbf{P}$-certificate π^l at round $\lfloor j \rfloor_{n^l}$ (or else the statement proved by π^l would be different from the one that S_1 needs to provide a certificate about). The bracket operator simply "filters" out all messages that are generated at or after communication round $\lfloor j \rfloor_{n^{l-1}}$.*

As noted above, by construction, $\vec{\sigma}$ always satisfies the appropriate length restrictions. Thus, the only thing we need to ensure is that the certificates received by S_1 indeed prove the "right" statements for S_1 to be able to complete its WIUAs; we shall see why this is the case shortly.

We now turn to defining S_l for $2 \leq l \leq k+1$, inductively. Suppose $S_1, \ldots, S_{l-1}$ are defined. $S_l(1^n, x, \vec{M}, s, \ell)$ emulates the interaction among $S_{l-1}(1^n, x, \vec{M}, s, \ell), \ldots, S_1(1^n, x, \vec{M}, s, \ell), V^*$ for ℓ communication rounds, while expecting to receive external messages "on the left".

- In each communication round j with $j \mod n^{l-1} = 0$, after V^* sends a verifier message m_j, we distinguish two cases.
  - If $l = 2$, S_2 generates a certificate π^1_j (using P_{cert}) that $\mathit{wrap}(S_1(1^n, x, \vec{M}, s_j, j), V^*, \tau, j) = m_j$, where τ is the set of messages S_1 has received so far, and outputs $(j, 1, \pi^1_j)$.
  - If $l > 2$, S_l continues to emulate the round to the point that (the internally emulated) S_{l-1} outputs its message $(j, l-2, \pi^{l-2}_j)$, and then S_l generates a certificate π^{l-1}_j that $\mathit{wrap}'(S_{l-1}(1^n, x, \vec{M}, s_j, j), \tau, j) = \eta$, where τ is the set of messages that S_{l-1} has received so far and η is the set of messages S_{l-1} has generated so far (in the internal emulation). Then S_l outputs the message $(j, l-1, \pi^{l-1}_j)$.
- In each communication round j s.t., $j \mod n^l = 0$, after generating its message $(j, l-1, \pi^{l-1}_j)$, S_l expects to receive external messages $(j, l'-1, \pi^{l'-1}_j)$ "on the left" for every $l' > l$ such that $j \mod n^{l'-1} = 0$. S_l simply relays these messages to its internally emulated $S_{l-1}, \ldots S_1$.

Finally, S_l outputs its own view at the end of the execution (which in particular, contains the view of V^*, and all the messages generate by S_l).

Note that the construction of $S_2, \ldots, S_{k+1}$ ensures that S_1 will always have the appropriate certificates to complete every WIUA it reaches; as a consequence, S_1 never gets "stuck".

Let $\vec{S} = (S_1, \ldots, S_k)$. The final simulator $S(1^n, x)$ simply runs $S_{k'}(1^n, x, \vec{S}, s, T(n+|x|))$, where s is a uniformly random string of length n, $T(n+|x|)$ is a polynomial upper-bound on the number of messages sent by V^* on input 1^n and statement $x \in \{0,1\}^{\mathrm{poly}(n)}$, and $k' = \lceil \log_n T(n+|x|) \rceil + 1$, and then extracts and outputs the view of V^* from the output of $S_{k'}$. Note that since T is polynomial in n, k' is a constant.

Running-time of S We first note that essentially the same argument as for Protocol 1 shows that S_1 runs in polynomial time: It only takes S_1 polynomial-time to generate the commitments in Phase 1 (since V^* has a polynomial-length description, and the programs $\tilde{S}_l$'s have length polynomial in the size of V^*). During the WIUA in Phase 2, the length of the witness used by the simulator is polynomial in length of the programs $\tilde{S}_l$'s, and their inputs and outputs, all of which are polynomial in the circuit-size of V^*. Since the **P**-certificates verification time is polynomial in the length of the statement proved, it follows that the relation being proved in the WIUA has a time complexity that is upper bounded by a fixed polynomial in the length of V^*. By the relative prover efficiency condition of the WIUA, each such proof only requires some fixed polynomial-time, and thus the whole execution of S_1 takes some fixed polynomial time (in the size of V^* and thus also in the length of x.) It directly follows that also $\tilde{S}_1$'s running-time is polynomially bounded.

It now follows by an induction that S_l and thus $\tilde{S}_l$ run in polynomial time for every constant l. Suppose S_{l-1} and $\tilde{S}_{l-1}$ run in polynomial time. Since S_l is simply providing certificates about the execution of $\tilde{S}_{l-1}$, it follows by the relative prover-efficiency condition of **P**-certificates, that S_l runs in polynomial time, and thus also $\tilde{S}_l$. Finally, as S simply runs $S_{k'}$ with a constant k', the running-time of S is polynomially bounded as well.

Indistinguishability of the simulation Note that by construction of S, it follows that the simulation never gets "stuck" in the sense that whenever V^* expects a WIUA in some session, S has an appropriate "fake" witness and can complete the WIUA using this "fake" witness. Indistinguishability of the simulation follows in identically the same way as for Protocol 1.

4.3 Dealing with Randomized P-certificates

As mentioned above, to simplify the exposition, our protocol uses strong **P**-certificate system $(P_{\mathsf{cert}}, V_{\mathsf{cert}})$ with *deterministic* prover and verifier strategies. We here sketch how to deal with the case when P_{cert} and V_{cert} are randomized.

- **Handling randomized V_{cert}.** If V_{cert} is randomized, we simply need to the verifier V generate the randomness for V_{cert}, but to guarantee soundness of the **P**-certificate, V needs to do so after the **P**-certificates are determined. We do this by adding a new communication round before Phase 2 where the prover first is asked to commit to the k **P**-certificates $\pi^1, \ldots, \pi^k$ that it wants to use in Phase 2 (the honest prover should simply commit to $0^{k \cdot n}$) and next the verifier selects randomness $\rho^1, \ldots, \rho^k$ for V_{cert} for each of these certificates. In Phase 2, the prover is then asked to demonstrate that for each certificate $l \in [k]$, V_{cert} using randomness ρ^l accepts π^l.

- **Handling randomized P_{cert}.** If P_{cert} is randomized, the helper simulators $S_2, \ldots, S_{k+1}$ also become randomized. As with S_1, there is now a potential "randomness-dependent" issue since the simulators generate certificates about their own behaviour in earlier communication rounds (in particular, S_1 needs to know the randomness of all "helper" simulators). We can break the circularity by using forward secure PRGs in exactly the same way as was done for S_1; each the simulator S_l use *independent* seeds $s^{(l)}$ for a forward secure PRG to expand the randomness for generating level-$(l-1)$ certificates in each communication round, and then uses the seed $s_j^{(l)}$ as an input to $\tilde{S}_l$'s when generating certificates at communication round j.

4.4 A Note on Uniform Assumptions

We remark that even in the case of uniform soundness, our protocol currently relies on families of hash-functions collision-resistant also for non-uniform polynomial-time. Note, however, that for our soundness proof, it suffices to use commitment schemes that are binding for uniform polynomial-time algorithms and a WIUA where the proof of knowledge property is proven secure using a uniform security reduction. (We still need the hiding and the witness indistinguishability properties to hold for non-uniform polynomial-time to establish $\mathcal{ZK}$ with arbitrary auxiliary inputs). We see no obstacles in getting these properties by instantiating our protocol with statistically-hiding commitments and a "special-purpose" WIUA from [PR05a], which also relies on statistically-hiding commitments, but we haven't verified the details. In particular, if we only rely on statistically-hiding commitments where the (computational) binding hold against uniform polynomial-time algorithms, such commitment can be based on families of hash functions collision-resistant against uniform polynomial-time.

References

[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.

[BCCT12a] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCCT12b] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. Cryptology ePrint Archive, Report 2012/095, 2012. `http://eprint.iacr.org/`.

[BCPT12] Eleanor Birrell, Kai-Min Chung, Rafael Pass, and Sidharth Telang. Randomness-dependent message security. Unpublished Manuscript, 2012.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Computational Complexity*, pages 162–171, 2002.

[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.

[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.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[Bon03] Dan Boneh, editor. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BP04a] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2004.

[BP04b] 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*, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[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.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 595–613. Springer, 2009.

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

[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.

[CLP12] Ran Canetti, Huijia Lin, and Omer Paneth. Public coin concurrent zero-knowledge in the global hash model. Manuscript, 2012.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

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

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012.

[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.

[FGL$^+$91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *FOCS*, pages 2–12. IEEE Computer Society, 1991.

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

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GJ10] Vipul Goyal and Abhishek Jain. On the round complexity of covert computation. In Schulman [Sch10], pages 191–200.

[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/`.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *IACR Cryptology ePrint Archive*, 2011:456, 2011.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

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

[Gol93] Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *J. Cryptology*, 6(1):21–53, 1993.

[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.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[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.

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

[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.

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[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 Boneh [Bon03], pages 96–109.

[Pas03a] Rafael Pass. On deniability in the common reference string and random oracle model. In Boneh [Bon03], pages 316–337.

[Pas03b] 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.

[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.

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

[PTW11] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. *SIAM J. Comput.*, 40(6):1529–1553, 2011.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In *TCC*, pages 553–570, 2008.

[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.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sch10] Leonard J. Schulman, editor. *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 2010.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.