2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Constant-Round Concurrent Zero Knowledge From P-Certificates

Kai-Min Chung
Academia Sinica, Taiwan
chung@cs.cornell.edu

Huijia Lin
University of California, Santa Barbara
huijial@gmail.com

Rafael Pass
Cornell University
rafael@cs.cornell.edu

***Abstract*—We present a constant-round concurrent zero-knowledge protocol for NP. Our protocol relies on the existence of families of collision-resistant hash functions, and a new, but in our eyes, natural complexity-theoretic assumption: the existence of P-certificates—that is, "succinct" non-interactive proofs/arguments for P. As far as we know, our results yield the first constant-round concurrent zero-knowledge protocol for NP with an explicit zero-knowledge simulator based on *any* assumption.**

I. 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]).

Chung is supported by NSF Award CNS-1217821, NSF Award CCF-1214844 and Pass' Sloan Fellowship; work was done when being at Cornell.

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

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 $\mathbf{NP}$ 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. Note that it could very well be the case that all "classic" zero-knowledge protocols already are concurrent zero-knowledge; thus, simply assuming that those protocols are concurrent zero-knowledge yields an assumption under which constant-round concurrent zero-knowledge (trivially) exists—in essence, we are assuming that for every attacker a simulator exists. Furthermore, as we discuss in Section III, if we make strong "concurrent extractability" assumptions of the knowledge-of-exponent type [Dam91], [HT98], [BP04], concurrent zero-knowledge is easy to construct.[1] But such extractability assumptions also simply assume that for every attacker, a simulator ("the extractor") exists. In essence, rather than basing constant-round concurrent zero-knowledge on a hardness assumption, it is based on a "knowledge" assumption; that is, an assumption that is very similar in flavour to simply assuming that a protocol is zero-knowledge. The central question that we address in this paper is thus the following:

> *Can constant-round concurrent zero-knowledge be based on any (reasonable) complexity-theoretic* **hardness assumption?**

As an additional point, even under the above-mentioned strong "knowledge" assumptions, an explicit construction of the concurrent zero-knowledge simulator is not known—it is simply assumed that one exists. For some applications of zero-knowledge such as *deniability* (see e.g., [DNS04], [Pas03b]), having an explicit simulator is crucial. As far as we know, there are currently no assumptions (no matter how crazy) under which constant-round concurrent zero-knowledge with an explicit simulator is known.

In fact, even in the common reference string (CRS) model, there are no known constructions of constant-round concurrent zero-knowledge where the simulator does not "program" the CRS; such zero-knowledge protocols were referred to as *deniable zero-knowledge* in the CRS model in [Pas03b].[2] Indeed, as shown in [Pas03b], the black-box lower-bounds for concurrent zero-knowledge in the plain model extend also to such a "non-programmable" CRS model.

[1]Furthermore, as shown in the recent independent work of [GS12], even a "non-concurrent" (but quite strong in a different way) extractability-type assumption can be used.

[2]Again, if the simulator gets to program the CRS, such a simulator cannot be used to get deniability.

A. *Our Results*

In this work, we present new complexity-theoretic assumptions, which in our eyes are both natural and reasonable (and can be efficiently falsified), under which constant-round concurrent zero-knowledge is achievable. Furthermore, we provide an explicit zero-knowledge simulator.

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

We may consider the existence of $\mathbf{P}$-certificates either in the "plain" model (without any set-up), or with some set-up, such as the CRS model. We may also consider various different notions of soundness: *uniform computational soundness*—which states that no *uniform* polynomial-time algorithm can output an accepting certificate for any false statement, *non-uniform computational soundness*—where the same condition holds also w.r.t. non-uniform polynomial-time attackers, and *statistical soundness*—where soundness condition holds also with respect to unbounded attackers restricted to selecting statements of polynomial length.

Note that in the plain model, non-uniform soundness and statistical Sundanese are equivalent, since if an accepting proof of a false statement exists, a non-uniform efficient attacker can simply get it as non-uniform advice. In the CRS model, however, the notions are (seemingly) distinct.

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* $\mathbf{P}$-*certificates*.

On the Existence of $\mathbf{P}$-certificates In the plain model, a candidate construction of uniformly computationally-sound $\mathbf{P}$-certificate systems come from Micali's CS-proofs [Mic00]. These constructs provide short certificates even for all of $\mathbf{NEXP}$. However, since we here restrict to certificates only for $\mathbf{P}$, the assumption that these constructions are sound (resp. strongly sound) $\mathbf{P}$-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.[3]

In our eyes, on a qualitatively level, the assumption that Micali's CS-proofs yield strong $\mathbf{P}$-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 be efficiently checked, 2) no attacks are currently known, and 3) the "design-principles" underlying the construction rely on similar intuitions.

As a final point, recall that Micali's CS-proofs rely on the Fiat-Shamir heuristic, which in general may result in unsecure schemes [Bar01], [GK03]; however, note that whereas Micali's construction is *unconditionally* secure in the random oracle model, the counterexamples of [Bar01], [GK03] extensively rely on the underlying protocol only being computationally secure; as such, at this time, we have no reason to believe that the Fiat-Shamir heuristic does not work for Micali's protocol (or any other protocol that is unconditionally secure in the random oracle model).

In the CRS model, we may additionally assume that Micali's CS-proofs satisfy *non-uniform* computational soundness. Additionally, several recent works provide constructions of "SNARGs" (succinct non-interactive arguments) for $\mathbf{NP}$ in the CRS model [Gro10], [Lip12], [BCCT13], [GGPR13]; such constructions are trivially $\mathbf{P}$-certificates with non-uniform computational soundness in the CRS model. However, since we restrict to languages in $\mathbf{P}$, checking whether soundness of any of these constructions is broken now becomes efficiently checkable (and thus assuming that they are secure becomes falsifiable).

Finally, let us remark that even statistically-sound $\mathbf{P}$-certificates may exist: Note that the existence of statistically-sound strong $\mathbf{P}$-certificates is implied by the assumption 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 [BLV06]. In essence, these assumptions 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,[4] 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 $\mathbf{P}$-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 circuits, and the existence of a strong $\mathbf{P}$-certificate system with uniform (resp. non-uniform) soundness. Then there exists a constant-round concurrent zero-knowledge argument for $\mathbf{NP}$ with uniform (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).*

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 following Rogaway [Rog06])[5], 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 $\mathbf{P}$-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 $\mathbf{P}$ implemented using some specific hash function (e.g., SHA-256), then this attacker can be used to break this *particular* implementation of CS-proofs.

Furthermore, by the above argument, we may also instantiate our protocol with P-certificates in the CRS model, leading to a constant-round concurrent zero-knowledge protocol (with non-uniform soundness) in the non-programmable CRS model.

Beyond Concurrent $\mathcal{ZK}$ Since the work of Barak [Bar01], non-black-box simulation techniques have

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

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

[5]Rogaway used this argument to formalize what it means for a concrete hash function (as opposed to a family of hash functions) to be collision resistant.

been used in several other contexts (e.g., [BGGL01], [DGS09], [BP12], [Lin03], [PR03a], [Pas04], [BS05], [GJ10]. We believe that our techniques will be applicable also in those scenarios. In particular, in the full version, we show that our protocols directly yield a constant-round simultaneously-resettable $\mathcal{ZK}$ [BGGL01], [DGS09] for $\mathbf{NP}$, and discuss applications to concurrent secure computation.

II. Proof Outline

In this extended abstract, due to lack of space, we only provide a (detailed) outline of our techniques. The full proof is found in the full version [CLP12]. The starting point of our construction is Barak's [Bar01] non-black-box zero-knowledge argument for $\mathbf{NP}$. Let us start by very briefly recalling the ideas behind his protocol (following a slight variant of this protocol due to [PR03b]). Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\mathrm{poly}(n)}$, the Prover P and Verifier V, proceed in two stages. In Stage 1, P starts by sending a computationally-binding commitment $c \in \{0,1\}^n$ to 0^n; V next sends a "challenge" $r \in \{0,1\}^{2n}$. In Stage 2, P shows (using a witness indistinguishable argument of knowledge) that either x is true, or there exists a "short" string $\sigma \in \{0,1\}^n$ such that c is a commitment to a program M such that $M(\sigma) = r$.[6]

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,
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* $\mathbf{P}$*-certificates*—$\mathbf{P}$-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 $\mathbf{P}$-certificates about the generation of $\mathbf{P}$-certificates etc., in a

[6]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 $h(m)$.

tree-like fashion.

Step 1: Bounded Concurrency Using P-certificates In this first step, we present a (somewhat convoluted) protocol using strong $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-certificates π; the rest of the computation has *a-priori* bounded complexity (depending only on the size and running-time of V^*). To take advantage of this observation, we thus have the simulator only commit to a program that generates prover messages (in identically the same way as the actual simulator), but getting certificates $\vec{\pi}$ as input.

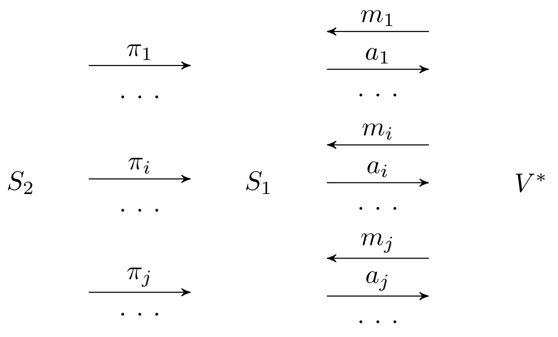

Figure 1. Simulation using $\mathbf{P}$-certificates.

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^*.

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.[7] S_1 next simulates Stage 2 by checking if it has received a message (j, π_j) in the left interaction, where j is the communication round (in the right interaction with V^*) where the verifier sends its random challenge and expects to receive the first message of Stage 2; if so, it uses $M_1 = \tilde{S}_1$ (and the randomness it used to commit to it), j and σ being the list of messages received by S_1 in the left interaction, as a "fake" witness to complete Stage 2.

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

The actual simulator S just runs S_2 and recovers from

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

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 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.[8]

Step 2: Full Concurrency using Unique $\mathbf{P}$-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 $\mathbf{P}$-certificates—that is, $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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) $\mathbf{P}$-certificates Let us finally see how to implement the above idea while using "plain" (i.e., non-unique) $\mathbf{P}$-certificates. The above protocol is no longer sound 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 generated and again there is no *a-prior* upper bound on the number of such certificates, so it seems we haven't really gained

[8]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 [CLP13].

anything.

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.

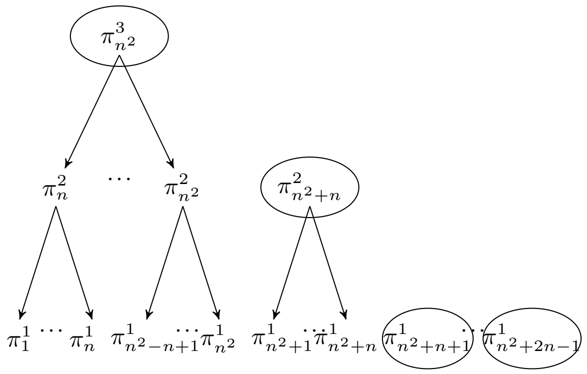

Figure 2. An illustration of the tree structure for generating $\mathbf{P}$-certificates. Nodes that are not circled are "certified" certificates; nodes that are circled are "dangling" certificates.

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 $\mathbf{P}$-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^*.

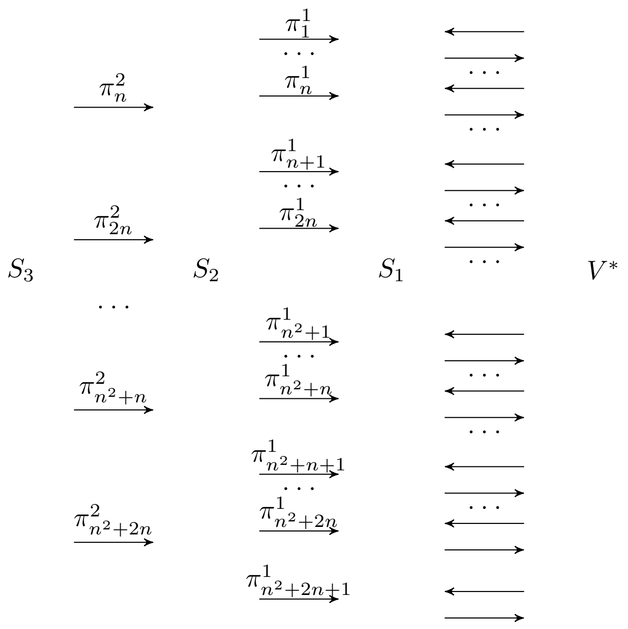

Figure 3. Simulation using second-level $\mathbf{P}$-certificates.

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.

III. Related Work

We provide a 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. As we elaborate upon in the full version, this can be formalized as a "concurrent" extractability assumption of the "knowledge-of-exponent"-type [Dam91]. (As shown in [Pas03b], any sufficiently length-expanding random oracle satisfies exactly such an concurrent extractability assumption—this was used in [Pas03a] to construct a concurrent $\mathcal{ZK}$ protocol in the "non-programmable" random oracle model.)

A very recent work by Gupta and Sahai [GS12] independent of the current work presents an alternative "non-concurrent" (but more structured) extractability assumption under which constant-round concurrent zero-knowledge can be achieved. One important difference between the above approaches and our work is that we here provide an *explicit* concurrent $\mathcal{ZK}$ simulator. The above-mentioned approaches simply *assume* 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 $\mathbf{P}$-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 [CLP13] constructs a *public-coin* concurrent zero-knowledge protocol using non-black-box simulation techniques[9]. 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 [CLP13] 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. 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: [CLP13] 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 $\mathbf{P}$-certificates to overcome the blow-up and obtain a constant-round protocol.

A recent work by Bitansky, Canetti, Chiessa, Tromer [BCCT13] present techniques for composing SNARKs (succinct non-interactive arguments of knowledge) for $\mathbf{NP}$; roughly speaking, [BCCT13] 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"

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

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 $\mathbf{NP}$ (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 $\mathbf{P}$-certificates are only used to demonstrate that lower-level $\mathbf{P}$-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 $\mathbf{P}$-certificates on a falsifiable assumption.

A. 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!

REFERENCES

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

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

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

[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.

[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.

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

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

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

[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.

[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] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from falsifiable assumptions. Cryptology ePrint Archive, Report 2012/563, 2012. http://eprint.iacr.org/.

[CLP13] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

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

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

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

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

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

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

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, pages 626–645, 2013.

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

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

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

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

[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.

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

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

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

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

[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.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm 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.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.

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

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

[Pas03a] Rafael Pass. On deniability in the common reference string and random oracle model. In *CRYPTO*, pages 316–337, 2003.

[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 multiparty 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.

[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.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In *VIETCRYPT*, pages 211–228, 2006.

[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.

[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.