How to Go Beyond the Black-Box Simulation Barrier

Boaz Barak
Department of Computer Science, Weizmann Institute of Science
Rehovot, ISRAEL
boaz@wisdom.weizmann.ac.il

Abstract

The simulation paradigm *is central to cryptography. A* simulator *is an algorithm that tries to simulate the interaction of the adversary with an honest party, without knowing the private input of this honest party. Almost all known simulators use the adversary's algorithm as a* black-box. *We present the first constructions of non-black-box simulators. Using these new non-black-box techniques we obtain several results that were previously proven to be impossible to obtain using black-box simulators.*

Specifically, assuming the existence of collision resistent hash functions, we construct a new zero-knowledge argument system for $\mathbf{NP}$ *that satisfies the following properties:*

1. *This system has a constant number of rounds with negligible soundness error.*

2. *It remains zero knowledge even when composed concurrently* n *times, where* n *is the security parameter.*

   Simultaneously obtaining 1 and 2 has been recently proven to be impossible to achieve using black-box simulators.

3. *It is an Arthur-Merlin (public coins) protocol.*

   Simultaneously obtaining 1 and 3 was known to be impossible to achieve with a black-box simulator.

4. *It has a simulator that runs in* strict *polynomial time, rather than in* expected *polynomial time.*

   All previously known constant-round, negligible-error zero-knowledge arguments utilized expected *polynomial-time simulators.*

1. Introduction

The *simulation paradigm* is one of the most important paradigms in the definition and design of cryptographic primitives. For example, this paradigm occurs in a setting in which two parties, Alice and Bob, interact and Bob knows a secret. We want to make sure that Alice hasn't learned anything about the secret as the result of this interaction, and do so by showing that Alice could have *simulated the entire interaction by herself*. Therefore, she has gained no further knowledge as the result of interacting with Bob, beyond what she could have discovered by herself.

The canonical example of the simulation paradigm is its use in the definition of *zero knowledge proofs*, as presented by Goldwasser, Micali and Rackoff [20]. Suppose that both Alice and Bob know a public graph G, and in addition Bob knows a hamiltonian cycle C in this graph. In a zero knowledge proof, Bob manages to *prove* to Alice that the graph G contains a hamiltonian cycle, and yet Alice has learned nothing about the cycle C, as she could have simulated the entire interaction by herself.

A crucial point is that we do not want Alice to gain knowledge even if she *deviates arbitrarily* from the protocol when interacting with Bob. This is usually formalized in the following way: for every algorithm V^* that represents the strategy of the verifier (Alice), there exists a simulator M^* that can simulate the entire interaction of the verifier and the honest prover (Bob) without access to the prover's auxiliary information (the hamiltonian cycle).

Consider the simulator's task even in the easier case in which Alice does follow her prescribed strategy. One problem that the simulator faces is that in general it is impossible for it to generate a convincing proof that the graph G is hamiltonian, without knowing a hamiltonian cycle in the graph. How then can the simulator generate an interaction that is indistinguishable from the actual interaction with the prover? The answer is that the simu-

Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)
0-7695-1390-5/02 $17.00 © 2002 IEEE

lator has two advantages over the prover, which compensate for the serious disadvantage of (the simulator's) not knowing a hamiltonian cycle in the graph. The first advantage is that unlike in the true interaction, the simulator has access to the verifier's random tape. This means that it can actually determine the next question that the verifier is going to ask. The second advantage is that, unlike in the actual interaction, the simulator has many attempts at answering the verifier's questions. This is because if it fails, it can simply choose not to output this interaction but rather retry again and output only the take in which it succeeds. This is in contrast to an actual proof, where if the party attempting to prove failed even once to answer a question then the proof would be rejected. The difference is similar to the difference between a live show and a taped show. This second technique is called *rewinding* because the simulator who fails to answer a question posed by the verifier, simply rewinds the verifier back and tries again.

Most of the known zero knowledge protocols make use of this rewinding technique in their simulators. However this technique, despite all its usefulness, has some problems. These problems arise mainly in the context of parallel and concurrent compositions. For example, using this technique it is impossible to show that a constant-round zero-knowledge proof[1] is closed under concurrent composition [8]. It seems also very hard to construct a constant-round zero-knowledge proof with a simulator that runs in *strict* polynomial time (rather than *expected* polynomial running time) or a constant-round proof of knowledge with a strict polynomial time knowledge extractor. Indeed, no such proofs were previously known.

The reason that all the known simulators were confined to the rewinding technique is that it is very hard to take advantage of the knowledge of the verifier's random tape when using the verifier's strategy as a *black-box*. Let us expand a little on what we mean by this notion. As noted above, to show that a protocol is zero knowledge, one must show that a simulator exists for *any* arbitrary algorithm V^* that represents the verifier's strategy. Almost all the known protocols simply used a *single* simulator that used the algorithm V^* as an oracle (i.e. as a black-box). Indeed it seemed very hard to do anything else, as using V^* in any other way seemed to entail some sort of "reverse-engineering" that is considered a very hard (if not impossible) thing to do.

It can be shown that for black-box simulators, the knowledge of the verifier's random tape does not help the simulator, because a verifier can have its randomness "hardwired" into its algorithm (for instance in the form of a description of a hash/pseudorandom function). Therefore, black-box simulators are essentially restricted to using the rewinding technique, and so suffer from its consequences. Indeed, several negative results have been proved about the power of black-box simulators, starting with the results of Goldreich and Krawczyk [18] regarding non-existence of *black-box* 3-round zero knowledge proofs and constant-round Arthur-Merlin zero-knowledge proofs, to the recent result of Canetti, Kilian, Petrank and Rosen [8] regarding impossibility of black-box constant-round concurrent zero-knowledge.

We show that the belief that one can not construct non-black-box simulators is not true. That is, given the code of a (possibly cheating) efficient verifier as an auxiliary input, the simulator may significantly use this code in other ways than merely running it, and so obtain goals that are provably impossible to obtain when using the verifier only as a black-box. Specifically, assuming the existence of collision resistent hash functions[2], we construct a new zero-knowledge argument (i.e., a computationally sound proof) for any language in $\mathbf{NP}$ that satisfies the following properties:

1. It has a constant number of rounds and negligible soundness error.
2. It remains zero knowledge if executed concurrently n times, where n is the security parameter. We call a protocol that satisfies this property a *bounded concurrent zero-knowledge* protocol[3] (the choice of n is quite arbitrary and could be replaced with any *fixed* polynomial (e.g. n^3) in the security parameter.)
3. It is an Arthur-Merlin (public coins) protocol.
4. It has a simulator that runs in *strict* polynomial time, rather than *expected* polynomial time.
5. It is actually an argument of knowledge.

The above protocol should be contrasted with the following impossibility results regarding *black-box* zero-knowledge arguments: Goldreich and Krawczyk have shown that obtaining Properties 1 and 3 together is impossible to achieve when using black-box simulation.[4] In addition, the proofs of Canetti, Kilian, Petrank and Rosen [8] show that no constant-round protocol is

[1] Here and throughout this paper, we only consider zero knowledge proofs or arguments that have negligible soundness error.

[2] Actually in this paper we need to require that there exist hash functions that are collision resistent against adversaries of size that is some fixed "nice" super-polynomial function (e.g., $n^{\log n}$ or $n^{\log\log n}$). This requirement can be relaxed as is shown in [1].

[3] This is in contrast to a standard concurrent zero-knowledge protocol [10] that remains zero-knowledge when executed concurrently any polynomial number of times.

[4] This is not the only reason that the existence of constant-round Arthur-Merlin zero-knowledge arguments is interesting. Subsequently to this paper, [2] showed that the existence of such protocols has several implications in the *resettable* model.

bounded concurrent zero-knowledge (i.e. satisfies Property 2) with a black-box simulator.

Moreover, this is the first zero-knowledge argument that achieves both Properties 1 and 4. The previous state of affairs, where one needed to allow *expected* polynomial time simulators in order to have constant-round zero-knowledge arguments, was rather unsatisfactory and resolving it was suggested as an open problem [12, Chap. 3], [16, Sec. 4.12.3].

A Brief Summary of Related Work. Zero-knowledge proofs were introduced by Goldwasser, Micali and Rackoff in [20]. Goldreich, Micali and Wigderson [19] gave a zero-knowledge proof for any language in $\mathbf{NP}$. Constant-round zero-knowledge arguments and proofs were first presented by Feige and Shamir [13], Brassard , Crépeau and Yung [5], and Goldreich and Kahn [17].

A non-black-box zero-knowledge argument was suggested by Hada and Tanaka [21]. However, there it was under a non-standard assumption that in itself was of a strong "reverse-engineering" flavor.

Concurrent zero-knowledge was defined by Dwork, Naor and Sahai [10]. They also constructed a concurrent zero knowledge protocol in the timing model. Richardson and Kilian constructed a concurrent zero-knowledge argument (in the standard model) with polynomially many rounds [27]. This was later improved by Kilian and Petrank to polylogarithmically many rounds [24]. As mentioned above, Canetti *et al* [8] (improving on [25] and [28]) show that this is essentially the best one can hope for a protocol using *black-box* simulation.

Some of our techniques (the use of CS proofs) were first used in a similar context by Canetti, Goldreich and Halevi [7]. CS proofs were defined and constructed by Kilian [22], [23] and Micali [26]. Dwork, Naor, Reingold and Stockmeyer [9] have explored some connections between the existence of non-black-box 3-round zero-knowledge arguments and other problems in cryptography. Some other "non-black-box results" were shown by Barak *et al* [3] in the context of code obfuscation.

2. Overview of Our Construction

To give a flavor of our techniques, we present in this section a zero-knowledge argument system for $\mathbf{NP}$ satisfying all the properties mentioned in the introduction, other than being bounded concurrent zero-knowledge and an argument of knowledge.[5] That is:

1. The protocol has a constant number of rounds and negligible soundness error.
2. The protocol is an Arthur-Merlin (public coins) protocol.
3. The protocol has a simulator that runs in *strict* polynomial time, rather than *expected* polynomial time.

Items 1 and 2 together imply that the simulator for this protocol *must* be a non-black-box simulator, because if $\mathbf{NP} \not\subseteq \mathbf{BPP}$ then no such protocol can be black-box zero-knowledge [18]. Therefore, this protocol is *inherently* a non-black-box zero-knowledge argument.

Structure of this Section. Our construction uses a general paradigm, which first appeared in a paper by Feige, Lapidot and Shamir [11], called the FLS paradigm. This paradigm is described in Section 2.1. The FLS paradigm gives us a framework into which we need to plug two components in order to obtain a zero-knowledge protocol. The first component, called a generation protocol, is described in Section 2.2. The construction described there involves some novel ideas, which account for the difference between this zero-knowledge argument and previously known protocols. In Section 2.3 we describe the second component that we need to plug into the framework: this is a witness indistinguishable proof that has some special properties. Our construction in Section 2.3 uses previous constructions of Kilian [22] and Micali [26]. In Section 2.4 we review the protocol obtained by plugging the two components into the framework, and sketch why it is indeed a zero-knowledge argument with the required properties.

Notation For a binary relation R, we let $L(R) \overset{def}{=} \{x|\exists w \text{ s.t. } (x,w) \in R\}$. A string w such that $(x,w) \in R$ is called a *witness* or a *solution* for x. In the context of a proof system for a language L, we may refer to a string x as a *theorem*, where by proving the theorem x we mean proving the statement "$x \in L$".

2.1. The FLS Paradigm

Our construction follows the well-known paradigm, introduced by Feige, Lapidot and Shamir, which we call the FLS paradigm[6]. In the FLS paradigm, we convert a witness indistinguishable[7] proof into a zero-knowledge proof by providing the simulator with a *fake witness*.

[5] The protocol, described in the full version of this paper, that satisfies all 5 properties is a modified version of the protocol described in this section.

[6] This paradigm has been introduced in the context of non-interactive zero-knowledge, but has been used also in the interactive setting (e.g. [27]).

[7] For the definition of witness indistinguishable proofs see [14] or [16, Sec. 4.6].

This fake witness allows the simulator to simulate a proof, where the witness indistinguishable property is used to ensure that the verifier cannot tell the difference between the real interaction and the simulated one. Of course, one must ensure that no polynomially bounded prover will be able to obtain such a fake witness, or else the resulting system will no longer be sound.

To be more concrete, the protocol will consists of two phases. In the first phase, called the *generation phase*, the prover and verifier will generate some string, denoted τ. The prescribed prover and verifier both *ignore* their input in performing this stage (and so it can be performed even before knowing what is the theorem that will be proved). In the second stage, called the *WI proof phase*, the prover proves (using a witness indistinguishable proof system) that *either* the theorem is true or that the generated string τ satisfies some property (in our case the property will be that τ is a member of a particular language $L(S)$, for some fixed relation S)[8]. For technical reasons we actually need to use a (witness indistinguishable) *proof of knowledge* for the second phase, rather than just a (WI) proof of membership. Also, for our purposes it is enough to use an argument (i.e., a computationally sound proof) rather than a (statistically sound) proof. It should be noted that if both the generation protocol and the WI proof of knowledge are constant-round Arthur-Merlin protocols, then so will be the resulting zero-knowledge argument.

The generation phase will be designed such that the following holds:

1. On the one hand, if the verifier follows its prescribed strategy, then it is *guaranteed* that the string τ does not satisfy the above property (e.g. $\tau \notin L(S)$), or at least that no polynomial-size prover will be able to prove that τ does satisfy the property.

2. On the other hand for *any* (possibly cheating) verifier V^*, there exists a simulator M^* that is able to simulate V^*'s interaction with the (honest) prover during the generation protocol in such a way that not only will the generated string τ satisfy the property, but the simulator M^* will be able to prove that this is the case. As our property will be membership in some language $L(S)$, this means that the simulator will "know" a witness σ such that $(\tau, \sigma) \in S$. This string σ is the *fake witness* that allows the simulator to simulate the second stage.

Previous protocols that used a similar framework employed relatively simple generation protocols, and the simulators used *rewinding* to obtain the solution σ for the generated string τ (see for example [27]). In particular, no previous generation protocol was a constant-round Arthur-Merlin protocol, as such a protocol would necessarily have a non-black-box simulator (because otherwise one could have plugged such a protocol into the FLS framework (with a constant-round Arthur-Merlin WI proof of knowledge) and obtain a constant-round Arthur-Merlin protocol that is black-box zero-knowledge).

2.2. Our Generation Protocol

Our generation protocol is a constant-round Arthur-Merlin protocol, and so has a non-black-box simulator. Before presenting the protocol itself, we describe more precisely the conditions that such a protocol should satisfy. For some fixed relation S we require that the protocol satisfies that:

1. (Hardness) The prescribed verifier has the property that for *any* (possibly cheating) polynomial-size prover P^*, for τ denoting the string generated by the interaction of the prescribed verifier with P^*, the probability that P^* outputs a solution for τ (i.e., a string σ such that $(\tau, \sigma) \in S$) is negligible.
2. (Easiness) For *any* (possibly cheating) polynomial-size verifier V^*, we require that there exists a simulator M^* that outputs a pair (e, σ) that satisfies the following:
   (a) The first element e is a simulation of the view of V^* in a real execution with the (honest) prover. That is, the view of V^* when interacting with the prescribed prover is computationally indistinguishable from the first element, e. Recall that the *view* of V^* consists of all the messages V^* receives and its random tape. Without loss of generality we assume that this view contains also all messages that V^* sends and so also contains the string τ generated in this execution.
   (b) For τ denoting the string generated in an execution with view e, the string σ is a solution for τ (i.e., $(\tau, \sigma) \in S$).

2.2.1 The basic approach

The general approach in our generation protocol is to try to set up a situation where finding a solution to the generated string τ will be equivalent to predicting one of the verifier's messages before it is sent. We will then obtain the hardness condition by having the prescribed verifier choose this message at random, and so infer that

[8] At this point the reader may think of S as an $\mathbf{NP}$ relation, however in our actual construction we will use a relation S of slightly higher complexity.

no prover can predict this message with non-negligible probability. In contrast, for *any* (possibly cheating) verifier V^*, the corresponding simulator, knowing (i.e. getting) V^*'s code and random tape *can* predict the messages that V^* sends.

In this subsection, we will construct a generation protocol for which the easiness property holds only with respect to verifiers that, when run with security parameter 1^n, the total length of their code (where by *code* we mean the description of the verifier's Turing machine along with any auxiliary input that the verifier gets) and random tape is at most n^3. We call such verifiers n^3-*bounded* (the choice of n^3 is quite arbitrary). We will later modify the protocol in order to remove this restriction.

Consider a two-round protocol in which the prover first sends a message z and then the verifier responds with a message v. We know that for any (possibly cheating) verifier V^*, the message v is obtained by applying the *next-message function*[9] of V^* , denoted NM_{V^*}, to the message z; that is, $v = NM_{V^*}(z)$. Given the code and random tape of V^*, one can construct in polynomial-time a program that computes the function NM_{V^*}. For n^3-bounded verifiers, the length of this program will be at most n^3, where 1^n is the security parameter (and so we can assume without loss of generality that the length of this program is exactly n^3).

In a protocol of the above type, we say that the message z *predicts* the message v if z is a commitment to a program Π such that $\Pi(z) = v$. That is, z predicts v if there exists a program Π and a string s (serving as coins for the commitment) such that $z = \mathcal{C}(\Pi; s)$ and $\Pi(z) = v$, where $\mathcal{C}$ is some fixed (perfect-binding and computationally-hiding) commitment scheme. We see that for any z, if the message v is chosen uniformly and independently of z in $\{0,1\}^n$, then the probability that $v = \Pi(z)$, where $\Pi = \mathcal{C}^{-1}(z)$ [10], is at most 2^{-n}.

These observations lead us to the following suggestion for a generation protocol:

Protocol 1:

- Input: 1^n - security parameter

1. Prover's first step (P1): Compute $z \leftarrow \mathcal{C}(0^{n^3})$. Send z.
2. Verifier's first step (V1): Choose v uniformly in $\{0,1\}^n$. Send v.

We let the string τ generated by the protocol be its transcript (i.e., $\tau = \langle z, v\rangle$). We define the following relation S_1: for $\tau = \langle z, v\rangle$ and $\sigma = \langle \Pi, s\rangle$ we say that $(\tau, \sigma) \in S_1$ if and only if:

1. z is a commitment to Π using coins s. That is, $z = \mathcal{C}(\Pi; s)$.
2. $\Pi(z) = v$.

Clearly, Protocol 1 and the relation S_1 satisfy the hardness property. In fact they satisfy this property even with respect to *computationally unbounded* provers, because (regardless of the prover's strategy) as long as the verifier follows its prescribed strategy, it will be the case that $\tau \notin L(S_1)$ with probability at least $1 - 2^{-n}$. Protocol 1 for S_1 also satisfies the easiness property with respect to n^3-bounded verifiers: Let V^* be such a verifier. The simulator M^* for V^* will compute z to be a commitment (with coins s) for a program Π that computes next message function NM_{V^*} (the simulator will compute such a program Π of length n^3 using the code and random tape of V^*). By the security of the commitment scheme, we know that the distribution of z is computationally indistinguishable from the distribution of the first message of the prescribed prover. Moreover, if we let v be V^*'s reply to z (i.e. $v = NM_{V^*}(z)$) then we know that it holds that $(\langle z, v\rangle, \langle \Pi, s\rangle) \in S_1$. We see that we have a simulator that satisfies both Parts (a) and (b) of the easiness condition. The following observation will be useful for us in the future:

Observation 2.1. *For any n^3-bounded verifier V^*, the simulator M^* defined above actually satisfies a stronger condition than is needed for Part (b) of the easiness condition: not only that it holds that $\Pi(z) = v$, but that if V^* runs in time $t(|z|)$, then on input z, the program Π outputs v* within $t(|z|)^2$ steps.

Observation 2.1 implies that the fact that the simulator M^*'s second output σ is a solution for τ can be verified in time which is some fixed polynomial in the running time of V^*. From now on we shall make this requirement, that the simulator outputs an *efficiently verifiable solution*, a part of the easiness condition.

2.2.2 Difficulties and Resolving them

We have two problems with the construction above:

1. Our simulator works only for n^3-bounded verifiers (i.e., verifiers for which the total size of the code and randomness is at most n^3). In particular this means that when we plug this generation protocol into the FLS framework, we will obtain a protocol that is zero-knowledge in some meaningful sense but *not* with respect to verifiers with (polynomial-size) auxiliary input. This is less desirable, because the condition of zero knowledge with respect to auxiliary input is needed for many applications (e.g., for proving preservation of zero-knowledge with respect to sequential composition).

[9] Note that once we fix the random tape of V^*, this is a (deterministic) function.

[10] That is, Π is the unique string committed to by z

2. As defined above, the relation S_1 is undecidable (this can be seen via a direct reduction from the halting problem). This is a more serious problem, because it implies that when plugging our generation protocol into the FLS framework, we will need to use a witness indistinguishable proof for undecidable relations (which does not exist[11]).

The reason for the first problem is that the simulator's first message is a commitment to a program of about the same size as the code and random tape of the verifier, and a perfectly binding commitment cannot reduce the size of its input. Our solution will be to use a collision-resistent hash function $h(\cdot)$, and so have the simulator send a commitment to a *hash* of the program, instead of the program itself. By combining a commitment with a hash function we obtain a commitment scheme that is only *computationally* binding , and so the hardness condition will now hold only against adversaries of bounded computational power. As we want the hash function to be collision-resistent also for non-uniform adversaries, we will need to use a hash function *ensemble* $\{h_\kappa\}_{\kappa\in\{0,1\}^*}$, and so we will have the verifier send the index κ of the hash function to be used as a preliminary step (we assume that h_κ is a function from $\{0,1\}^*$ to $\{0,1\}^{|\kappa|}$). Also, for technical reasons we require that the hardness condition will hold against $n^{\log n}$-size adversaries (rather than just against polynomial-size adversaries), and so we will assume that the hash function remains collision resistent also against $n^{\log n}$-size adversaries[12].

To deal with the second problem, we will change the definition of the relation S_1 to require that on input z, the program Π outputs v *within* $n^{\log\log n}$ *steps.* This will reduce the complexity of S_1 to being an $\mathbf{Ntime}(n^{\log\log n})$ relation. As a first observation, note that this modification does not harm the hardness condition, because for any string τ, we only eliminate some of the solutions σ that were allowed by the previous definition. This modification also does not harm the easiness condition: The reason is that, due to Observation 2.1, the solution $\sigma = \langle \Pi, s\rangle$ outputted by the simulator for a string $\tau = \langle z, v\rangle$ will always satisfy that $\Pi(z)$ outputs v within the time complexity of the simulated verifier V^*, and so within less than $n^{\log\log n}$ steps. We will also need to change the definition of S_1 in order to accommodate the hash function. We denote this modified relation by S.

Generation Protocol

- Input: 1^n - security parameter

1. Verifier's first step (V1): Choose $\kappa \leftarrow_{\mathrm{R}} \{0,1\}^n$. Send κ.
2. Prover's first step (P1): Compute $z \leftarrow \mathcal{C}(0^n)$. Send z.
3. Verifier's first step (V2): Choose $v \leftarrow_{\mathrm{R}} \{0,1\}^n$. Send v.

The string outputted by this protocol is its transcript $\tau = \langle \kappa, z, v\rangle$

Definition of relation S: For $\tau = \langle \kappa, z, v\rangle$ and $\sigma = \langle \Pi, s\rangle$ we say that $(\tau,\sigma) \in S$ if and only if:

1. z is a "hashed commitment" to Π using coins s and hash function h_κ. That is, $z = \mathcal{C}(h_\kappa(\Pi); s)$.
2. On input z, the program Π outputs v within $n^{\log\log n}$ steps.

Figure 1. Definitions for the generation protocol and the relation S

The definitions of the $\mathbf{Ntime}(n^{\log\log n})$ relation S and the generation protocol are shown in Figure 1. We claim that this relation and generation protocol satisfy both the easiness and the hardness condition. Observe that the hardness condition holds against $n^{\log n}$-size provers, rather than just against polynomial-size provers (we will later make use of this fact). Observe also that the generation protocol is a constant-round Arthur-Merlin protocol.

2.3. A Witness Indistinguishable Argument for $\mathbf{Ntime}(n^{\log\log n})$

Suppose that S was an $\mathbf{NP}$ relation. In this case, if we plug the generation protocol of Figure 1 into the FLS framework, then we can use any constant-round Arthur-Merlin WI proof of knowledge for $\mathbf{NP}$ for the second phase, and obtain a zero-knowledge argument system for $\mathbf{NP}$.

Our problem is that the relation S is *not* an $\mathbf{NP}$ relation, but rather an $\mathbf{Ntime}(n^{\log\log n})$ relation. A natural question is whether a WI argument system (with polynomial communication) for $\mathbf{Ntime}(n^{\log\log n})$ relations exists, and if so, does there exist such a system that is a constant-round Arthur-Merlin protocol. The answer to

[11] This is not entirely accurate, as one might be able to use a WI proof system with some guarantees regarding *instance based complexity* (as opposed to worst-case complexity). In fact, our final approach will be along these lines.

[12] For the sake of simplicity, in this overview section we fix the hardness of the hash function ensemble to be $n^{\log n}$. Actually, as is shown in the full version of this paper, the same analysis works whenever the hardness is a "nice" super-polynomial function (e.g., $n^{\log\log n}$). In [1] it is shown that a (slightly more complicated) protocol can be constructed under the more standard assumption that a hash function ensemble exists with arbitrary super-polynomial hardness.

the first question is yes: Kilian [23] has constructed, under suitable complexity assumptions, a constant-round WI argument system for $\mathbf{Ntime}(n^{\log\log n})$ relations. However, his system is not an Arthur-Merlin protocol[13], and so we will need to construct such a system ourselves. In fact we will need an "enhanced" WI argument for $\mathbf{Ntime}(n^{\log\log n})$, that satisfies some extra properties, such as *prover efficiency* and *verifier efficiency* (as defined by Micali in [26]). Let S' be an $\mathbf{Ntime}(n^{\log\log n})$ relation. The properties that we will require (and our construction will fulfill) from a WI argument system for S' will be the following (where Properties 1-3 are the standard definition of WI arguments):

1. (Witness Indistinguishability) For any τ and σ, σ' such that $(\tau, \sigma), (\tau, \sigma') \in S'$, and for any polynomial-size verifier V^*, the view of V^* when interacting with the honest prover that gets σ as auxiliary input is computationally indistinguishable from its view when interacting with the honest prover that gets σ' as auxiliary input.
2. (Perfect Completeness) The honest prover, when given as input $(\tau, \sigma) \in S'$, convinces the prescribed verifier with probability 1 that indeed $\tau \in L(S')$.
3. (Computational Soundness) For any polynomial-size strategy P^*, and any $\tau \notin L(S')$, the probability that P^* can convince the prescribed verifier is negligible.
4. (Verifier Efficiency) The length of all messages and the running time of the prescribed verifier are some fixed polynomial in the security parameter
5. (Prover Efficiency) There is a fixed Turing machine $M_{S'}$ that decides S', such that the honest prover, on input $(\tau, \sigma) \in S'$, runs in time which is some fixed polynomial in the running time of $M_{S'}$ on (τ, σ). That is, the time to prove in the WI system is some fixed polynomial in the time to deterministically verify the solution for τ.
6. (Weak Proof of Knowledge) For any polynomial-size prover strategy P^* and any input τ, there exists a $n^{O(\log\log n)}$-time knowledge extractor E such that the probability that P^* can convince the prescribed CS verifier that $\tau \in L(S')$ is polynomially related to the probability that $E(\tau)$ outputs a solution to τ[14].

Sketch of our construction. The main tool that we use to construct our WI argument system is *CS proofs* (CS stands for computationally sound). A CS proof system is a proof system that satisfies Properties 2-6 above (that is, everything apart from the witness indistinguishable property). CS Proofs (by a somewhat different definition) were defined and constructed by Kilian [22] and Micali [26] based on the $\mathbf{PCP}$ Theorem. It follows from [22] and [26] that under our complexity assumptions (namely the existence of collision-resistent hash functions strong against $n^{\log n}$ size adversaries) there exists a 4-round Arthur-Merlin CS proof system for any $\mathbf{Ntime}(n^{\log\log n})$ relation. We denote the 4 messages sent in the CS proof by $\alpha, \beta, \gamma, \delta$. We assume without loss of generality that $|\alpha| = |\beta| = |\gamma| = |\delta| = n^2$.

Let S' be an $\mathbf{Ntime}(n^{\log\log n})$ relation. Our construction for a WI argument system for S', described in Figure 2, will consist of two phases. In the first phase, the prover and verifier engage in an "encrypted" CS proof that $\tau \in L(S')$, when τ is the theorem that is to be proved. Specifically, the verifier follows the strategy of the CS verifier (i.e., sends random strings of length n^2), but the prover sends only *commitments* to the actual CS prover's messages. In the second phase, the prover proves (using a standard[15] constant-round WI proof of knowledge for $\mathbf{NP}$) that the transcript consisting of the verifier's messages along with the decommitments of the prover's messages, would have convinced the prescribed CS verifier that $\tau \in L(S')$.

2.4. Plugging Everything In

Let $L = L(R)$ be an $\mathbf{NP}$ language. We claim that if we plug the generation protocol and the relation S of Figure 1 along with the WI argument of Figure 2 (applied to the relation S' defined such that $\langle x, \tau\rangle \in L(S')$ if $x \in L(R)$ or $\tau \in L(S)$) into the FLS framework, then we obtain a constant-round Arthur-Merlin zero-knowledge argument for L (with a simulator that runs in strict polynomial time). We will mention some issues that are involved in proving this statement:

- The fact that the honest prover runs in polynomial time is proven using the *prover efficiency* condition of the WI argument system. This follows from the fact that when proving that $x \in L$, the honest prover gets as auxiliary input a witness w such that $(x, w) \in R$. As R is an $\mathbf{NP}$ relation, checking whether $(x, w) \in R$ can be done in time which is some fixed polynomial in $|x|$, say $|x|^c$ for some

[13] This is not surprising, since his protocol is in fact a black-box zero-knowledge argument system.

[14] Note that we do not require that the probability of E's success in outputting a solution will be the same or close to the probability of P^* convincing the verifier. We only require that these probabilities are polynomially related. Note also that we allow the knowledge extractor to run in time $n^{O(\log\log n)}$ rather than just in polynomial time.

[15] Actually we will need to require that the WI proof for $\mathbf{NP}$ satisfies a slightly stronger condition called *strong* witness indistinguishablity (see [16], Section 4.6.1.1). All standard WI proof satisfy this condition, because it is implied by zero-knowledge and it is closed under parallel and concurrent composition.

- Common Input: τ (the theorem to be proven in $L(S')$), 1^n: security parameter
- Auxiliary Input to prover: σ such that $(\tau, \sigma) \in S'$.

1. *Phase 1: "Encrypted" CS proof.*
   (a) Verifier's first step (V1): Choose α uniformly in $\{0,1\}^{n^2}$; Send α.
   (b) Prover's first step (P1):
      i. Initiate CS prover algorithm with (σ, τ).
      ii. Feed α to the CS prover, and obtain its response β. ($\beta \in \{0,1\}^{n^2}$)
      iii. Send a commitment to β: Choose s' uniformly in $\{0,1\}^{n^3}$; Compute $\hat{\beta} \leftarrow \mathcal{C}(\beta; s')$; Send $\hat{\beta}$.
   (c) Verifier's second step (V2): Choose γ uniformly in $\{0,1\}^{n^2}$; Send γ;
   (d) Prover's second step (P2):
      i. Feed γ to the CS prover, and obtain its response δ. ($\delta \in \{0,1\}^{n^2}$)
      ii. Send a commitment to δ: Choose s'' uniformly in $\{0,1\}^{n^3}$; Compute $\hat{\delta} \leftarrow \mathcal{C}(\delta; s'')$; Send $\hat{\delta}$.
2. *Phase 2: An* **NP** *WI Proof of Knowledge.*
   (a) Prover and verifier engage in a constant-round Arthur-Merlin witness indistinguishable proof of knowledge that $\langle \tau, \alpha, \hat{\beta}, \gamma, \hat{\delta} \rangle \in L(T)$ where T is the following **NP** relation[a]:
   $(\langle \tau, \alpha, \hat{\beta}, \gamma, \hat{\delta} \rangle, \langle \beta, s', \delta, s'' \rangle) \in T$ if and only if the following holds:
      i. $\hat{\beta}$ is a commitment to β using coins s'. That is, $\hat{\beta} = \mathcal{C}(\beta; s')$.
      ii. $\hat{\delta}$ is a commitment to δ using coins s''. That is, $\hat{\delta} = \mathcal{C}(\delta; s'')$.
      iii. The CS transcript $(\tau, \alpha, \beta, \gamma, \delta)$ convinces the CS verifier that $\tau \in L(S')$.

[a] The fact that T is indeed an **NP** relation is implied by the *verifier efficiency* condition of the CS proof. This condition implies that membership in T can be verified in polynomial time.

Figure 2. A Witness Indistinguishable Argument for $L(S')$

$c > 0$. Yet the *prover efficiency* condition implies that proving that either $x \in L(R)$ or $\tau \in L(S)$ (i.e. proving that $\langle x, \tau \rangle \in L(S')$) using the WI argument system with auxiliary input w takes time which will be some fixed polynomial in $|x|^c$.

- Computational soundness is proven using the *(weak) proof of knowledge* condition of the WI argument and the *hardness* condition of the generation protocol: By the proof of knowledge condition of the WI argument, any prover that convinces the verifier that $x \in L$ "knows" either a witness w such that $(x, w) \in R$ or a witness σ such that $(\tau, \sigma) \in S$ (where τ is the string generated in the generation phase). Yet the hardness condition of the generation protocol implies that the probability of the latter event is negligible. This means that if the prover convinces the verifier with non-negligible probability that $x \in L$, then the prover must "know" a witness w such that $(x, w) \in R$ and in particular it must be the case that such w exists and so indeed $x \in L$.

  Note that we use the fact that the hardness condition holds against $n^{\log n}$-size adversaries, because the knowledge extractor of our WI argument does not run in polynomial time, but rather in $n^{O(\log \log n)}$ time.

- Zero-knowledge is proved using the *witness indistinguishability* condition of the WI argument and the *easiness* condition of the generation protocol. The simulator for the zero-knowledge protocol uses the simulator of the easiness condition to simulate the first phase and then uses the solution σ provided by this simulator as an auxiliary input (i.e., a fake witness) to the prover strategy for the WI argument. The simulation will be computationally indistinguishable from a real interaction of the first phase by Part (a) of the easiness condition. The string σ will be a solution to the string τ generated in this simulation by Part (b) of the easiness condition.

  The prover efficiency condition for the WI argument, along with the condition stated in Observation 2.1 (i.e., that the simulator for the generation protocol outputs an efficiently verifiable solution), assures us that the simulator for the zero-knowledge protocol will run in (strict) polynomial time. This follows from the same reasons (described above) that the honest prover for the zero-knowledge argument runs in polynomial time[16]

[16] Note that, in contrast to the prover that runs in some fixed polynomial time, the simulator will run in time which is some fixed polynomial in the running time of the verifier that is being simulated.

Intuitively, the simulator presented above does not use the *rewinding* technique (although it is probably impossible to give a precise meaning to the phrase "not using the rewinding technique" for a simulator that gets as input the description of the code of the verifier). Rather, it uses the code and random tape of the verifier as a "fake" witness that allows it to simulate the behavior of the honest prover, that gets as input a real witness.

3. Conclusions and Future Directions

Arguably, our simulator does not "reverse-engineer" the verifier's code, although it applies some non-trivial transformations (such as a **PCP** reduction and a Cook-Levin reduction) to this code. Yet, we see that even without doing "true" reverse-engineering, one can achieve results that are impossible in a black-box model. This is in a similar vein to [3], where the impossibility of code obfuscation is shown without doing "true" reverse-engineering. One may hope to be able to define a model for an "enhanced black-box" simulator that would be strong enough to allow all the techniques we used, but weak enough to prove impossibility results that explain difficulties in constructing certain objects. We're not optimistic about such attempts.

There are several negative results regarding the power of *black-box* zero-knowledge arguments. The existence of non-black-box simulators suggests a reexamination of whether these negative results holds also for general (non-black-box) zero-knowledge. Indeed, we have already shown in this paper that some of these results *do not* hold in the general setting. The case of concurrent composition is an important example. The results of [8] imply that (for a constant-round protocol) it is impossible to achieve even *bounded* concurrency using black-box simulation. We have shown that this result does not extend to the non-black-box settings. However, it is still unknown whether one can obtain a constant-round protocol that is (fully) concurrent zero-knowledge. This is an important open question.

In this work we constructed our protocols based on the assumption that collision resistent hash function exist with some fixed "nice" super-polynomial hardness. It would be nicer to construct the protocol using the more standard assumption that collision resistent hash functions exist with arbitrary super-polynomial hardness. Indeed this can be done, and in [1], Barak and Goldreich construct CS proofs (by a slightly different definition) based on the more standard assumption. Using this construction, they show that a slightly modified version of the protocol presented here, enjoys the same properties under the more standard assumption.

Following this work, Barak, Goldreich, Goldwasser and Lindell [2] have shown another case where a black-box impossibility result does *not* hold in the general setting. Using results of the current paper, they construct an *argument of knowledge*[17] that is zero-knowledge in the *resettable* model. As noted by [6], this is trivially impossible in the black-box model.

There are also several negative results regarding what can be achieved using black-box *reductions* between two cryptographic primitives (rather than black-box use of an adversary by a simulator). Although this paper does not involve the notion of black-box reductions (and non-black-box reductions such as [19, 13] are already known to exist), the results here may serve as an additional sign that in general, similarly to relativized results in complexity theory, black-box impossibility results cannot serve as strong evidence toward real world impossibility results.

The fact that we've shown a constant-round Arthur-Merlin zero-knowledge protocol, can be viewed as some negative evidence on the soundness of the Fiat-Shamir heuristic [15]. This heuristic converts a constant-round Arthur-Merlin identification scheme into a non-interactive signature scheme by replacing the verifier's messages with a hash function.[18] It is known that the resulting signature scheme will be totally breakable if the original protocol is zero-knowledge [9]. Thus, there exist some constant-round Arthur-Merlin protocols on which the Fiat-Shamir heuristic cannot be applied.

We've shown the first constant-round zero-knowledge protocol with a *strict* polynomial-time simulator, instead of an *expected* polynomial-time simulator. Another context in which *expected* polynomial time arises is in constant-round zero-knowledge proofs *of knowledge* (e.g., [12, Chap. 3], [16, Sec. 4.7.6.3]). In [4] Barak and Lindell construct, using the protocol presented here, a constant-round zero-knowledge argument of knowledge with a *strict* polynomial-time extractor.

Acknowledgments

First and foremost, I would like to thank Oded Goldreich. Although he refused to co-author this paper, Oded's suggestions, comments, and constructive criticism played an essential part in the creation of this work. I would also like to thank Alon Rosen, Shafi Goldwasser and Yehuda Lindell for very helpful discussions. In particular, it was Alon's suggestion to use the FLS paradigm to construct the zero-knowledge argument, a

[17] Here and in [4] we use a slightly relaxed definition of arguments of knowledge, that allows the knowledge extractor access to the description of the prover, rather than limiting it to using the prover as an oracle.

[18] This heuristic is usually applied to 3 round protocols, but it can be applied to any constant-round Arthur-Merlin protocol.

change which considerably simplified the construction and its presentation.

References

[1] B. Barak and O. Goldreich. Cs proofs under a standard assumption. In preparation, 2001.

[2] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. These proceedings., 2001.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahay, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. To appear in *CRYPTO 2001*, 2001.

[4] B. Barak and Y. Lindell. Zero-knowledge arguments of knowledge with strict polynomial-time extraction. In preparation, 2001.

[5] G. Brassard, C. Crépeau, and M. Yung. Everything in NP can be argued in *perfect* zero-knowledge in a *bounded* number of rounds. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*, pages 192–195. Springer-Verlag, 1990, 10–13 Apr. 1989.

[6] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge (extended abstract). In ACM, editor, *Proceedings of the 32nd annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 235–244. ACM Press, 2000.

[7] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, 23–26 May 1998.

[8] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. Record 2001/051, Cryptology ePrint Archive, June 2001. An extended abstract appeared in STOC01.

[9] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. In IEEE, editor, *40th Annual Symposium on Foundations of Computer Science: October 17–19, 1999, New York City, New York,*, pages 523–534. IEEE Computer Society Press, 1999.

[10] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 409–418, New York, May 23–26 1998. ACM Press.

[11] Feige, Lapidot, and Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SICOMP: SIAM Journal on Computing*, 29, 1999.

[12] U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. PhD thesis, Weizmann Institute of Science, 1990.

[13] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–545. Springer-Verlag, 1990, 20–24 Aug. 1989.

[14] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In ACM, editor, *Proceedings of the 22nd annual ACM Symposium on Theory of Computing, Baltimore, Maryland, May 14–16, 1990*, pages 416–426, 1990.

[15] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proc. CRYPTO'86*, pages 186–194. LNCS 263, Springer Verlag, 1986.

[16] O. Goldreich. *Foundations of Cryptography*, volume 1 – Basic Tools. Cambridge University Press, 2001.

[17] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, Summer 1996.

[18] O. Goldreich and H. Krawczyk. On the composition of Zero-Knowledge Proof systems. *SICOMP*, 25(1):169–192, 1996. Preliminary version appeared in ICALP90, pages 268–290.

[19] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing Machinery*, 38(3):691–729, July 1991.

[20] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[21] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. Cryptology ePrint Archive, Report 1999/009, 1999. `http://eprint.iacr.org/`.

[22] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In ACM, editor, *Proceedings of the 24th annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 4–6, 1992*, pages 723–732, 1992.

[23] J. Kilian. Improved efficient arguments (preliminary version). In D. Coppersmith, editor, *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 311–324. Springer-Verlag, 27–31 Aug. 1995.

[24] J. Kilian and E. Petrank. Concurrent zero-knowledge in poly-logarithmic rounds. Cryptology ePrint Archive, Report 2000/013, 2000. `http://eprint.iacr.org/`, an extended abstract appeared in STOC01.

[25] J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the Internet. In IEEE, editor, *39th Annual Symposium on Foundations of Computer Science: proceedings: November 8–11, 1998, Palo Alto, California*, pages 484–492, 1998.

[26] S. Micali. CS proofs. In S. Goldwasser, editor, *Proceedings: 35th Annual Symposium on Foundations of Computer Science, November 20–22, 1994, Santa Fe, New Mexico*, pages 436–453. IEEE Computer Society Press, 1994.

[27] Richardson and Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1999.

[28] A. Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO: Proceedings of Crypto*, 2000.