On Constant-Round Precise Zero-Knowledge

Ning Ding and Dawu Gu

Department of Computer Science and Engineering
Shanghai Jiao Tong University, China
{dingning,dwgu}@sjtu.edu.cn

Abstract. Precise zero-knowledge, introduced by Micali and Pass [STOC'06], captures the idea that a view of any verifier can be *indifferently* reconstructed. Though there are some constructions of precise zero-knowledge, constant-round constructions are unknown to exist. This paper is towards constant-round constructions of precise zero-knowledge. The results of this paper are as follows.

- We propose a relaxation of precise zero-knowledge that captures the idea that with a probability *arbitrarily polynomially* close to 1 a view of any verifier can be *indifferently* reconstructed, i.e., there exists a simulator (without having $q(n), p(n,t)$ as input) such that for *any* polynomial $q(n)$, there is a polynomial $p(n,t)$ satisfying with probability at least $1 - \frac{1}{q(n)}$, the view of any verifier in every interaction can be reconstructed in $p(n,T)$ time by the simulator whenever the verifier's running-time on this view is T. Then we show the impossibility of constructing constant-round protocols satisfying our relaxed definition with all the known techniques.
- We present a constant-round precise zero-knowledge argument for any language in **NP** with respect to our definition, assuming the existence of collision-resistant hash function families (against all $n^{O(\log\log n)}$-size circuits).

1 Introduction

(*Question.*) Zero-knowledge proofs were introduced by Goldwasser, Micali and Rackoff [6]. Their definition essentially states that an interactive proof of $x \in L$ provides zero (additional) knowledge if, for any efficient verifier V, the view of V in the interaction can be "indistinguishably reconstructed" by an efficient simulator S-interacting with no one- on just input x. Since efficiency is formalized as polynomial-time, a worst-case notion, zero-knowledge too automatically becomes a worst-case notion. Micali and Pass [7] argued that this worst-case definition of zero-knowledge may not suffice to characterize that the view can be reconstructed *indifferently.* For instance, taking part in an interaction to gain a view only requires n steps while the simulator may need n^{10} steps to reconstruct this view. It is hard to say n and n^{10} are indifferent.

Hence [7] put forward a notion of *precise zero-knowledge*, which augments the definition of zero-knowledge by presenting an additional precision requirement that a prover provides a zero-knowledge proof of $x \in L$ if the view v of any verifier in an interaction with the prover about x can be reconstructed in the same time within a constant/polynomial factor.

To construct precise zero-knowledge protocols [7] developed a method, called the "cut-off" technique. A simulator with the "cut-off" technique still adopts the rewind strategy, but in the first run it records the verifier's running-time and then in the second run if the verifier cannot output the message within this recorded time then the simulator cancels this rewind. Though its success probability (in extracting secret) in each rewind becomes smaller, after $\omega(1)$ or $\omega(\log n)$ rewinds the simulator can succeed with overwhelming probability, achieving polynomial or linear precision respectively. Recently, [3] proposed a notion of precise time and space simulatable zero-knowledge which strengthes the notion of precise zero-knowledge by requiring that a view of any verifier in each interaction with a prover can be reconstructed in the same time and space simultaneously. Then it adopted an improved "cut-off" technique to construct some precise time and space simulatable zero-knowledge protocols.

Also, [8] presented a slightly relaxed notion of *weak-precise zero-knowledge* that requires the precision requirement holds with overwhelming probability. This paper does not distinguish the two notions for concision of statement.

There are also some negative results. [7] showed there don't exist black-box precise zero-knowledge protocols for any non-trivial language, and [8] showed Barak's non-black-box zero-knowledge argument [1] are imprecise due to the imprecise simulation strategy.

As the known precise zero-knowledge protocols use at least $\omega(1)$ rounds, a natural question arose at that time *if there exist constant-round precise zero-knowledge proofs or arguments*. As we will point out later, it seems unhopeful to adopt the known simulation strategies to realize precise simulation in constant-round constructions. So a feasible way towards this question is to present a slight but meaningful relaxation and then construct constant-round precise zero-knowledge protocols with respect to this relaxation. In this paper we will investigate the question in this way and attempt to present an answer to it.

1.1 Our Results

Recall that precise zero-knowledge requires that with probability $1 - \mathsf{neg}(n)$ a view of any verifier can be reconstructed in $p(n, T)$ time whenever the verifier's running-time on this view is T. We consider a relaxation that for a protocol to be precise zero-knowledge, there is a strict polynomial-time simulator (without having $q(n), p(n, t)$ as input) such that for *any* polynomial $q(n)$, there is a precision $p(n, t)$ satisfying with probability $1 - \frac{1}{q(n)}$ the simulator's running-time in outputting a view is bounded by $p(n, T)$ whenever a verifier's running-time on this view is T. Namely, we slightly relax the satisfiable probability from $1 - \mathsf{neg}(n)$ to $1 - \frac{1}{q(n)}$ where $q(n)$ can be an arbitrarily large polynomial.

Though a precise zero-knowledge protocol w.r.t. the standard definition in [7,8] of course satisfies our relaxation, our focus in this work is constant-round constructions. As we will show later, all known constant-round zero-knowledge protocols do not satisfy the relaxation. We will also point out that it seems still unhopeful to adopt the known simulation techniques to construct constant-round zero-knowledge protocols satisfying the relaxation.

Our main contribution in this paper is a constant-round construction of precise zero-knowledge protocols for any language in **NP** satisfying the relaxation. Formally, we achieve the following result.

Theorem 1. *Assume the existence of hash function families which collision-resistance hold against all $n^{O(\log\log n)}$-size circuits. Then there exists a constant-round precise zero-knowledge argument for each language in* **NP** *with respect to our relaxation.*

Our Technique. Notice that using the "cut-off" technique, we can achieve the following result that for any $q(n)$, a simulator with the "cut-off" technique can extract secret information (in one rewind interval) with probability $1-\frac{1}{q(n)}$ and thus the simulation satisfies a precision related to $p(n,t)$. Note that $q(n)$ should be known to the simulator in advanced. However, for any other polynomial $q'(n)$ this simulator may not achieve a precision (related to $q'(n)$) with probability $1-\frac{1}{q'(n)}$.

Our idea for going beyond this barrier is to rewind the verifier sufficiently large polynomial (less than $n^{\log\log n/5}$) times such that for any polynomial $q(n)$, we can always be ensured that the extraction can succeed with probability at least $1-\frac{1}{q(n)}$, while on the same time the whole simulation is still of strict polynomial-time (not super-polynomial-time).

To realize this idea, we combine the constructions of Barak's protocol [1] and ordinary zero-knowledge protocols to propose a desired protocol. The protocol consists of two phases. Phase 1 adopts a mixed structure of Barak's protocol and ordinary zero-knowledge protocols. Phase 2 is a constant-round zero-knowledge universal argument (ZKUA) of the statement that $x \in L$ or the transcript τ of phase 1 is in a language Λ. Basically, τ consists of the verifier's description and those messages generated in the ordinary run of one rewind interval. The language Λ consists of those τ's, in which if we perform at most $n^{\log\log n/5}$ rewinding runs with "cut-off" techniques at the verifier, it can output a correct secret information. (This actually means that we relegate all rewinds to the verification of $\tau \in \Lambda$.)

Our simulator for this protocol is an extension of Barak's simulator. That is, our simulator commits to (the oblivious machine of) the verifier's code and the auxiliary input bits *really accessed* by the verifier (not the entire auxiliary input) as well as performs other computation in phase 1. We remark that it is due to this strategy of committing only to accessed auxiliary bits that we can bypass the imprecision of Barak's simulator shown in [8]. In phase 2, the simulator employs a parallel simulation strategy which adopts the honest prover with the witness to interact with the verifier and in parallel calls the simulator of ZKUA. S finally adopts the output generated in that one of the two parallel simulation which first finishes as the simulated view of phase 2. We will show the simulator can achieve our definition indeed.

1.2 Organizations

The rest of the paper is arranged as follows. We assume familiarity with the notations and notions used throughout this paper. Section 2 analyzes the barriers for constructing constant-round precise zero-knowledge protocols. Section 3 formalizes our relaxation and demonstrates the impossibility of constructing constant-round protocols satisfying the relaxation with the known techniques. Section 4 presents our constant-round protocol and shows it is an argument for any language in **NP**. Section 5 presents a novel precise simulator for the protocol and proves that the protocol is precise zero-knowledge with respect to this relaxation. Section 6 concludes the paper.

2 Barriers for Constructing Constant-Round Precise Zero-Knowledge

In this section we introduce the notion of precise zero-knowledge in [7,8] and review the know constructions and lastly point out the barriers that all the known constructions and simulation strategies cannot realize constant-round precise zero-knowledge protocols.

2.1 Precise Zero-Knowledge

Counting Steps. If M is a probabilistic machine, denote by M_r the deterministic one obtained by fixing the content of M's random tape to r, by $\mathrm{STEPS}_{M_r(x)}$ the number of computational steps taken by M_r on input x.

Assume (P, V) uses u-round prover's messages. In an execution of (P, V), for any V^* with an auxiliary input aux, denote by $v = (x, \mathsf{aux}, (m_1, m_2, ..., m_u))$ the view of V^*, where m_i is the ith prover's message (w.l.o.g, assume V^* is deterministic). Then denote by $\mathrm{STEPS}_{V^*}(v)$ the number of computational steps taken by V^* running on view v, i.e. V^*'s running-time on input x and auxiliary input aux and letting the ith message received be m_i. In counting steps, we assume that an algorithm A, given the code of a second algorithm B and an input x, can simulate the computation of B on input x with linear-time slowdown [7]. (This assumption is inessential and we can use the logarithmic slowdown instead. Actually, for Turing machines and Random Access Machines, logarithmic slowdown is achieved.)

Definition 1. ***(Precise Zero-Knowledge [7,8])*** *Let (P, V) be an interactive proof or argument system for a language L, $p : N \times N \to N$ be a monotonically increasing 2-variate polynomial. We say that (P, V) is a zero-knowledge proof or argument with precision p if there exists a probabilistic algorithm S such that for every polynomial-time V^* and every auxiliary input $\mathsf{aux} \in \{0,1\}^*$ for V^*:*

1. *The view of V^* in an interaction with P, where the public input is x and P has a witness w for $x \in L$, is computationally indistinguishable from the output of $S(x, V^*, \mathsf{aux})$.*

2. *For sufficiently long random coins $r \in \{0,1\}^*$, let v be the view generated by $S_r(x, V^*, \mathsf{aux})$. Then* $\Pr[\mathrm{STEPS}_{S_r(x,V^*,\mathsf{aux})} \leq p(n, \mathrm{STEPS}_{V^*}(v))] = 1 - \mathsf{neg}(n)$.

We refer to S as above as a precise simulator. We say that (P, V) has polynomial precision or linear precision if $p(n,t)$ is a polynomial or linear function in t.

Remark 1. The original definition in [7,8] requires it holds with probability 1 that $\mathrm{STEPS}_{S_r(x,V^*,\mathsf{aux})} \leq p(n, \mathrm{STEPS}_{V^*}(v))$. Definition 1 is actually the definition of weak precise zero-knowledge in [8]. As said before, we don't distinguish the two notions in this paper.

2.2 Barriers for Achieving Constant-Round Constructions

Though there exist some constructions of precise zero-knowledge, constant-round constructions are unknown to exist. We now review the reasons that the known constructions cannot achieve the constant-round property and polynomial precision simultaneously.

Confliction Between "Cut-off" Simulation and Constant-Round Requirements. The known constructions of precise zero-knowledge in [7,8,3] employ "cut-off" simulation techniques. One such simulator counts the verifier's running-time in the first run and then in each rewinding run it uses the counted time to bound the verifier's computation, which finally ensures the simulation time is no more than a polynomial of the verifier's running-time. As shown in e.g. [8], the simulator's successful probability in extracting the secret information is at most $1 - 1/\mathrm{poly}(n)$ in a rewind interval. Thus to make the success probability overwhelming, a precise zero-knowledge protocol should use at least $\omega(1)$ rewind intervals. Thus it seems impossible to construct constant-round protocols with the "cut-off" techniques.

Imprecision of Barak's Non-black-box Simulator. One may ask if we can modify the known constant-round non-black-box zero-knowledge protocols to obtain precise zero-knowledge. However, the current situation is that the only known construction paradigm of constant-round non-black-box zero-knowledge is Barak's protocol [1]. However, as shown in [8], Barak's protocol is also imprecise. Recall that Barak's protocol consists of two phases. In phase 1, the verifier sends a random hash function h to the prover, which then responds with a commitment c to the 0-string. Then the verifier sends back a random string r. Let τ denote (h, c, r). In phase 2, the prover proves to the verifier via a WI universal argument that either $x \in L$ or $\tau \in \Lambda$ for a well-defined language $\Lambda \in \mathbf{Ntime}(n^{\log \log n})$. We now show the simulator for this protocol is imprecise.

Given access a verifier's code, Barak's protocol can be simulated without making use of rewinding: To perform simulation, the simulator commits to the hash of the verifier's message function including the auxiliary input (instead of committing to zeros). The verifier's next message function is then a program whose output, on input c, is r. This provides the simulator a valid witness to use in phase 2. However, consider a verifier V^* that has a very long auxiliary input, but most of the time only accesses a small portion of it. The simulator will always commit to the hash of whole description of V^* (including the whole auxiliary input) and will thus always take long time, while V^* might run fast a large portion of the time. Hence the simulation strategy is imprecise.

3 Our Relaxation

In this section we present a slight relaxation on the precision requirement and then point out that the known techniques cannot achieve this relaxation and the constant-round property simultaneously. A constant-round precise zero-knowledge protocol satisfying the relaxation will be introduced in next section.

3.1 The Relaxed Definition

Definition 1 requires that the precision p holds for all verifiers with probability $1-\mathsf{neg}(n)$. Thus a slight relaxation is to require that with a probability *arbitrarily polynomially* close to 1, there exists a precision p which holds for all verifiers. Namely, we only relax the satisfiable probability from $1-\mathsf{neg}(n)$ to $1-1/q(n)$ where $q(n)$ can be an arbitrarily large polynomial. The formal definition is as follows.

Definition 2. ***(The relaxation.)*** *Let (P, V) be an interactive proof or argument system for a language L. We say that (P, V) is a precise zero-knowledge proof or argument, if there exists a strict PPT algorithm S satisfying the following conditions:*

1. *For every poly-time V^* and every auxiliary input $\mathsf{aux} \in \{0,1\}^*$ to V^*, the output of $S(x, V^*, \mathsf{aux})$ is computationally indistinguishable from V^*'s real view interacting with $P(x, w)$.*
2. *For any polynomial $q(n)$, there is a monotonically increasing 2-variate polynomial $p : N \times N \to N$ such that for each poly-time V^* and each auxiliary input $\mathsf{aux} \in \{0,1\}^*$ to V^*, for sufficiently long random coins $r \in \{0,1\}^*$, letting v be the view generated by $S_r(x, V^*, \mathsf{aux})$, $\Pr[\mathrm{STEPS}_{S_r(x,V^*,\mathsf{aux})} \leq p(n, \mathrm{STEPS}_{V^*}(v))] \geq 1 - \frac{1}{q(n)}$.*

3.2 Limitations of the Known Techniques

At this moment one may ask if the known techniques for constructing precise zero-knowledge can achieve our relaxation and the constant-round property simultaneously. Unfortunately, we now point out that the known construction techniques cannot do this job indeed.

Let us first consider the known precise zero-knowledge protocols and the "cut-off" simulation techniques sketched in Section 2.2. As shown there, a simulator can succeed in extraction in each rewind interval with probability $1-\frac{1}{\mathrm{poly}(n)}$. For any polynomial $q(n)$, this success probability can be instantiated with $1-\frac{1}{q(n)}$.

Thus we have there is a polynomial $p(n, T)$ such that with probability $1 - \frac{1}{q(n)}$, the simulation is precise. However, in this construction $q(n)$ should be known to the simulator in advanced. That means for any larger polynomial $q'(n)$ this simulator cannot achieve any precision related to $q'(n)$ with probability $1 - \frac{1}{q'(n)}$. This shows Definition 2 cannot be satisfied by this construction.

Let us then consider Barak's protocol and his simulator. It can be seen that the barrier shown in Section 2.2 still exists w.r.t. out relaxation. That is, for a V^*, any $q(n)$ and p, there exists an aux of length more than $p(n, T)$ where T denotes V^*'s running-time. Since Barak's simulator needs to compute a commitment to V^*'s next message function which contains aux, its running-time is more than $p(n, T)$, contradicting condition 2 of Definition 2. So Barak's simulator is still imprecise with respect to Definition 2.

4 The Protocol

In this section we present a constant-round zero-knowledge argument. In Section 4.1 we give a high-level overview of the protocol. In Section 4.2 we present the actual construction and prove it is an interactive argument for **NP**.

4.1 The Overview

Our protocol consists of two phases. Phase 1 adopts a mixed structure of Barak's protocol and ordinary zero-knowledge protocols. The adopted structure of ordinary zero-knowledge protocols is a commitment-challenge-response paradigm. That is, the verifier first sends the prover n commitments to random strings which then responds with a random challenge indicating that some commitments should be revealed, and lastly the verifier opens the corresponding commitments.

The adopted structure of Barak's protocol is that in the above step of sending a challenge, the prover additionally sends the verifier a commitment to the 0-string and after the verifier opens some commitments, the prover sends the verifier a commitment to the hash of the 0-string. The goal of this strategy is for using Barak's non-black-box simulation strategy.

In phase 2, the prover proves to the verifier that it knows a witness for $x \in L$ or the transcript of phase 1 is in a language Λ. The definition of Λ requires that a transcript is in Λ iff what are committed by the prover in phase 1 are actually a program and some auxiliary input bits such that the program on given the auxiliary input bits and a challenge, can output the value of one unrevealed commitment by the verifier in phase 1.

4.2 Our Language Λ and Protocol

In this subsection we present our protocol, which uses the following cryptographic primitives:

- Let $\{\mathcal{H}_n\}_{n\in N}$, in which each $h \in \mathcal{H}_n$ maps $\{0,1\}^*$ to $\{0,1\}^n$, denote a hash function family which collision resistance holds against all $n^{O(\log\log n)}$-size circuit.
- Let HCom denote a two-round perfectly-hiding commitment scheme, BCom denote a one-round (or two-round) perfectly-binding commitment scheme. For simplicity of statement we always use C or Z to denote a perfectly-hiding or perfectly-binding commitment (while ignoring the possible first message of HCom and BCom).
- Let PRG denote a pseudorandom generator which admits the following properties. On input an n-bit random seed r, $\mathsf{PRG}(r)$ can iteratively generate arbitrarily polynomial pseudorandom bits. That is, it generates a fixed polynomial pseudorandom bits in each iteration, and then when run iteratively, $\mathsf{PRG}(r)$ can output an arbitrarily polynomial pseudorandom bits. [4] presented one construction of such PRGs. PRG will be used in our definition of language Λ. In the verification of membership in Λ, PRG will be run iteratively. That is, the verification sets up a repetition process and in each repetition $\mathsf{PRG}(r)$ is run once to output a fixed polynomial bits iteratively based on the internal state generated in the previous iterations.
- Let $\mathsf{CoinToss}$ denote a constant-round coin-tossing protocol, which runs as follows. The verifier first adopts HCom to compute a commitment to a random n-bit string and sends it to the prover, which responds with an independent n-bit strings. Then the verifier opens the commitment. Then the XOR of the two strings are the final coins.
- Let ZKUA denote a constant-round zero-knowledge universal argument for any language in **NE** defined in [2] and its simulator runs in strict polynomial-time. Note that [2] already presented a construction of ZKUA, but its simulator runs in expected polynomial-time. We can easily adapt it to our requitement. The ZKUA in [2] consists of two phases, where phase 1 is an "encrypted" commitment-challenge-commitment construction and phase 2 is an ordinary zero-knowledge protocol of knowledge, aiming at proving that the committed messages in phase 1 are consistent and satisfy the requirement of a PCP verification. Due to the call to the simulator of the ordinary zero-knowledge protocol in simulation, the simulator of this ZKUA runs in expected polynomial-time.

  Now we replace the zero-knowledge protocol in phase 2 by Barak's zero-knowledge protocol in [1] (note that phase 2 itself of Barak's protocol is a WI universal arument). Thus we can see the new ZKUA owns a strict polynomial-time simulator and still satisfies the weak proof of knowledge property. Thus the new ZKUA is a desired universal argument.

Now we present a definition of Λ in Definition 3. Assume L is an arbitrary **NP** language and our argument for L is shown in Protocol 1.

Definition 3. ***(Language*** Λ***).*** *We define* Λ *as follows:* $\tau = (h, \sigma, U, Z_1, Z_2, r) \in \Lambda$ *iff there exist a program* $\Pi \in \{0,1\}^n$*, a string* y *(as auxiliary input to* Π*) and coins* (s_1, s_2) *such that the following conditions can be verified within* $n^{\log\log n}$ *steps:*

Public input: $x \in \{0,1\}^n$ (statement to be proved is "$x \in L$");
Prover's auxiliary input: w (a witness that $x \in L$).

1. $V \to P$: Verifier selects $h \leftarrow_{\mathrm{R}} \mathcal{H}_n$. Choose $u_i \leftarrow_{\mathrm{R}} \{0,1\}^n$ and compute $C_i \leftarrow \mathsf{HCom}(u_i)$ for $1 \leq i \leq n$. Send h and $C_i, 1 \leq i \leq n$, to prover.
2. $P \to V$: Prover selects a random $\sigma \in \{0,1\}^n$ and computes $Z_1 \leftarrow_{\mathrm{R}} \mathsf{BCom}(0^n)$. Send σ and Z_1 to verifier.
3. $V \to P$: For each i satisfying that the ith bit of σ is 0, open all these C_i to prover.
4. $P \to V$: Prover computes $Z_2 \leftarrow_{\mathrm{R}} \mathsf{BCom}(h(0^n))$ and sends Z_2 to verifier.
5. $V \to P$: For all i satisfying the ith bit of σ is 1, send these u_i's to prover. Let U denote the set consisting of these u_i's. Then prove to prover via a constant-round zero-knowledge protocol of that these u_i's are the committed messages in the corresponding C_i's.
6. $P \leftrightarrow V$: Prover and verifier execute $\mathsf{CoinToss}$ to agree with random $r \in \{0,1\}^n$.

Let τ denote $(h, \sigma, U, Z_1, Z_2, r)$.

$P \to V$: Prover proves to verifier using its input w via ZKUA that $x \in L$ or $\tau \in \Lambda$.

Protocol 1. *Our precise zero-knowledge argument for L*

1. *$Z_1 = \mathsf{BCom}(\Pi; s_1)$ and $Z_2 = \mathsf{BCom}(h(y); s_2)$, where s_1, s_2 denote the randomness in commitments.*
2. *Run the following repetitions at most $n^{\log\log n/5}$ times. In the jth repetition, $j > 0$, do the following:*
   (a) *Iteratively run $\mathsf{PRG}(r)$ based on its existing internal state generated in the previous $(j-1)$ runs (if $j = 1$, PRG has no internal state) to generate sufficient pseudorandom bits, denoted (σ_j, r_j) (where $\sigma_j \in \{0,1\}^n$ and r_i is used as randomness in BCom, which length is a fixed polynomial that we omit specifying for clearness of statement);*
   (b) *Compute $Z_j^* \leftarrow \mathsf{BCom}(\Pi; r_j)$;*
   (c) *Run $\Pi(\sigma_j, Z_j^*, y)$ (y as auxiliary input) and during this running if Π needs an additional auxiliary input bit beyond y, cancel the running and denote by $\perp$ Π's output, else let U_j^* denote Π's output. In the case Π's output is U_j^*, if there exists at least one string in U_j^* (viewed as a set) such that it is in U, the verification of this condition succeeds. In other cases, continue the repetitions.*

A witness w for $\tau \in \Lambda$ is a tuple of (Π, y, s_1, s_2) satisfying Definition 3.

Theorem 2. *Assume the existence of hash function families which collision-resistance hold against all $n^{O(\log\log n)}$-size circuits. Then Protocol 1 is an interactive argument for L.*

Due to the hiding property of the commitments, any prover in the ordinary run cannot know the value of any unrevealed commitment and the committed program and auxiliary input by the prover are independent of the values in the unrevealed commitments. So the program cannot output any unrevealed message. Thus the transcript cannot belong to Λ, which leads to the soundness. We remark that the full proof of Theorem 2 employs the simulator of the zero-knowledge protocol in Step 5 to show the soundness. Due to short of space we omit the full proof.

5 The Precise Simulator

In this section we will prove the following result.

Theorem 3. *Protocol 1 is precise zero-knowledge with respect to Definition 2.*

In Section 5.1 we present the overview and actual description of our precise simulator. In Section 5.2 we show this simulator satisfies all requirements in Definition 2 and complete the proof of Theorem 3.

5.1 The Description

Let V^* denote any verifier of length $\{0,1\}^{n/2}$, aux be an arbitrarily long auxiliary input to V^*, S denote our simulator. Basically, S behaves like Barak's simulator, which runs as a prover interacting with V^*, tries to obtain a witness for the transcript τ of phase 1 and lastly uses this witness for the combined statement in phase 2. Informally, in phase 1 S commits to V^*'s next message function excluding aux in Step 2. W.l.o.g. let $\Pi \in \{0,1\}^n$ denote this committed program (more precisely, S first computes V^*'s next message program and then generates Π as an *oblivious machine* of this program). Notice that in Step 3 V^* (i.e. Π) may access some positions in aux. Thus S emulates Π's computation on input S's message of Step 2 as well as aux to generate an output, and at the same time records those auxiliary input bits Π really accesses. Let y denote the auxiliary bits. Then S commits to the hash of y in Step 4.

In phase 2 S adopts a parallel simulation strategy, in which it uses (Π, y) (as well as some coins used in commitments) as a witness for the transcript in Λ to interact with V^*, and in parallel, it employs the simulator of ZKUA to generate a view. S finally adopts that view generated in the one of the two parallel simulation which first finishes as the simulated view of phase 2.

Oblivious Machines. We outline some facts on obvious machines to help understand the execution of Π. A machine is oblivious if the sequence in which it accesses memory locations is equivalent for any two inputs with the same running time, e.g. oblivious Turing Machines (TM) and oblivious Random Access Machines (RAM). If an oblivious machine accesses more memory locations, it consumes more running-time. W.l.o.g. we assume a machine can be emulated by an oblivious machine with polynomial slowdown (for any unspecified appropriate computational model used for verifiers and the simulator). For instance, [9] showed how to emulate an arbitrary one-tape TM by a two-tape oblivious TM

```
Input: $x \in \{0,1\}^n$ (statement to be proved is "$x \in L$");
Verifier's code: $V^* \in \{0,1\}^{n/2}$; $V^*$'s auxiliary input: $\mathsf{aux} \in \{0,1\}^*$.

1. $V^* \to S$: $S$ emulates $V^*$ to output $h$ and $C_i$ for $1 \le i \le n$.
2. $S \to V^*$: Compute the oblivious machine corresponding to $V^*$'s next message
   function (excluding $\mathsf{aux}$), denoted $\Pi \in \{0,1\}^n$. $S$ selects a random $\sigma \in \{0,1\}^n$ and
   random coins $s_1$. Compute $Z_1 \leftarrow \mathsf{BCom}(\Pi; s_1)$ and send $(\sigma, Z_1)$ to $V^*$.
3. $V^* \to S$: $S$ emulates $\Pi$ to output some $u_i$'s. During the emulation, $S$ records those
   bits in $\mathsf{aux}$ accessed by $\Pi$. Denote by $y$ these accessed bits. (Also run $V^*$ to finish
   this step.)
4. $S \to V^*$: $S$ chooses random coins $s_2$ and computes $Z_2 \leftarrow \mathsf{BCom}(h(y); s_2)$. Send $Z_2$
   to $V^*$.
5. $V^* \to S$: $S$ emulates $V^*$ to output the remainder $u_i$'s and interacts with $V^*$ in the
   following zero-knowledge proof. Let $U$ denote the set consisting of these $u_i$'s.
6. $S \leftrightarrow V^*$: $S$ and $V^*$ run $\mathsf{CoinToss}$ to agree with coins $r \in \{0,1\}^n$.

Let $\tau$ denote $(h, \sigma, U, Z_1, Z_2, r)$.

$S \to V^*$: $S$ adopts the following parallel simulation strategy. It adopts the honest
prover's strategy with witness $(\Pi, y, s_1, s_2)$ to prove to $V^*$ via $\mathsf{ZKUA}$ that $x \in L$ or
$\tau \in \Lambda$, and in parallel it calls the simulator of $\mathsf{ZKUA}$ to generate a simulated view. $S$
halts whenever an arbitrary one of the two parallel simulation finishes, and adopts the
view in the finished one as the simulated view in phase 2.
```

Algorithm 1. *The precise simulator S*

with a logarithmic slowdown, and [5] showed how to emulate an arbitrary RAM by an oblivious RAM with a poly-logarithmic slowdown.

In our simulation, Π is an oblivious machine corresponding to the verifier's next message function. Thus in an execution of Π, more auxiliary input bits Π accesses, more running-time Π consumes. Thus that Π accesses more auxiliary input bits is equivalent to that Π consumes more running-time. So in the verification of $\tau \in \Lambda$, the condition that the execution of $\Pi(\cdots, y)$ should be canceled if Π needs to access more auxiliary input bits than y is equivalent to that the execution of $\Pi(\cdots, y)$ should be canceled if Π's running-time is more than (the poly-logarithmic overhead of) V^*'s running-time. This fact will be used to establish the precision property of S in next subsection.

The actual construction of the simulator is shown in Algorithm 1.

5.2 Analysis

In this subsection we present the following three claims to show that S can provide precise zero-knowledge property with respect to Definition 2. We also sketch the proofs of them but omit the full details due to short of space.

Claim 4. *For any polynomial $q(n)$, with probability at least $1-\frac{1}{q(n)}$ (Π, y, s_1, s_2) is a witness for $\tau \in \Lambda$ and on the occurrence of this the membership of $\tau \in \Lambda$ can be verified with (Π, y, s_1, s_2) in $O(q(n)\omega(\log n)T)$ time, where T denotes the running-time of $\Pi(\sigma, Z_1, y)$.*

Proof. (sketch) Let $\Pi(\cdots, y)$ denote Π with y hardwired. We need to show the conditions in Definition 3 can be satisfied. In the verification of $\tau \in \Lambda$, first assume each prover's message (σ_j, Z_j^*) is truly random. If in the execution of $\Pi(\sigma_j, Z_j^*, y)$ Π doesn't need to access any auxiliary bit beyond y, $\Pi(\sigma_j, Z_j^*, y)$ can output some decommitments to those in the verifier's message of Step 1. Since σ_j equals the real challenge with probability 2^{-n}, there is a commitment which was not revealed in the real interaction, but is revealed in the output of $\Pi(\sigma_j, Z_j^*, y)$. Thus the membership of $\tau \in \Lambda$ can be verified. Thus all that is left is to show Π doesn't need any auxiliary input bit beyond y in one repetition with high probability. In fact, it is true that within $2q(n)\omega(\log n)$ repetitions, the desired event occurs with probability at least $1-\frac{1}{2q(n)}-\mathsf{neg}(n)$. Now replacing each (σ_j, Z_j^*) by the pseudorandom strings output by $\mathsf{PRG}(r)$, the probability is at least $1-\frac{1}{q(n)}$. So the claim holds. □

Claim 5. *For any pair (x, w) such that w is the witness for $x \in L$, the view of V^* in an interaction with the honest prover of Protocol 1 holding (x, w) is computationally indistinguishable from S's output on input (x, V^*, aux).*

Proof. (sketch) We use S_1 (resp. S_2) to denote S's strategy with the first (resp. second) strategy used in phase 2. We have both S_1's and S_2's outputs are indistinguishable from the real view of V^*. In particular, over any noticeable sub probability space, S_1's output is indistinguishable from S_2's. We then show S's output is indistinguishable from S_1's. If there is only an negligible probability that S's strategy in phase 2 equals S_2, the fact holds. Otherwise, let B denote the noticeable event that S's strategy in phase 2 equals S_2. So on the occurrence of $\overline{B}$, S is actually S_1. Then for any D, $|\Pr[D^S(n)=1]-\Pr[D^{S_1}(n)=1]| \leq \Pr[\overline{B}]|\Pr[D^S(n)=1|\overline{B}]-\Pr[D^{S_1}(n)=1|\overline{B}]| + \Pr[B]|\Pr[D^S(n)=1|B]-\Pr[D^{S_1}(n)=1|B]| \leq |\Pr[D^S(n)=1|B]-\Pr[D^{S_1}(n)=1|B]|$. Since $|\Pr[D^S(n)=1|B]-\Pr[D^{S_1}(n)=1|B]| = \mathsf{neg}(n)$, S's output is indistinguishable from S_1's. The claim holds. □

Claim 6. *S satisfies condition 2 of Definition 2.*

Proof. (sketch) Choose a random sufficiently long coins *rand* for S. Let T' denote $\mathrm{STEPS}_{V^*_{\mathsf{aux}}}(v)$, where V^*_{aux} denotes V^* with the auxiliary input aux. We now analyze the running-time of S. First, S runs in strict polynomial-time. S's running-time for emulating V^*_{aux} is $O(T'+T)$. Due to Claim 4 and the relatively efficient prover property of ZKUA, S's running-time in phase 2 is $O(n^{c_0}T^c)$ for some constant c_0, c (where c_0, c are independent of T, T'). So we have there exists a polynomial $p(n, T')$ such that S's running-time is less than $p(n, T')$ with probability $1-\frac{1}{q(n)}$. The claim holds. □

Combining the three claims, we complete the proof of Theorem 3. Then combining it with Theorem 2, we also complete the proof of Theorem 1.

6 Conclusions

In this paper we investigate the question of how to construct constant-round precise zero-knowledge protocols. Since there are some barriers in solving this question that cannot be go beyond with all the known techniques, we look for a meaningful relaxation for precise zero-knowledge and a candidate constant-round construction with respect to the relaxation.

As a result, we propose one such relaxation that requires that with a probability arbitrarily polynomially close to 1, there exists a precision p such that the simulator can reconstruct all verifiers' views satisfying the requirement p. Then we show the impossibility of constructing constant-round protocols satisfying our relaxed definition with all the known techniques, which makes the relaxation meaningful with respect to constant-round constructions. The main contribution of this work is a constant-round precise zero-knowledge argument for **NP** satisfying the relaxation.

Acknowledgments. The authors are grateful to the reviewers of ICICS 2012 for their useful comments. This work is supported by the National Natural Science Foundation of China (61100209) and Shanghai Postdoctoral Scientific Program (11R21414500).

References

1. Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS, pp. 106–115 (2001)
2. Barak, B., Goldreich, O.: Universal arguments and their applications. In: IEEE Conference on Computational Complexity, pp. 194–203 (2002)
3. Ding, N., Gu, D.: Precise Time and Space Simulatable Zero-Knowledge. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 16–33. Springer, Heidelberg (2011)
4. Goldreich, O., Micali, S.: Increasing the expansion of pseudorandom generators (1996), `http://www.wisdom.weizmann.ac.il/oded/papers.html`
5. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM 43(3), 431–473 (1996)
6. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
7. Micali, S., Pass, R.: Local zero knowledge. In: Kleinberg, J.M. (ed.) STOC, pp. 306–315. ACM (2006)
8. Pass, R.: A precise computational approach to knowledge. Tech. rep., Ph. D. thesis, MIT. Available (2006)
9. Pippenger, N., Fischer, M.J.: Relations among complexity measures. J. ACM 26(2), 361–381 (1979)