Precise bounded-concurrent zero-knowledge proofs for NP

Ning DING* and DaWu GU

Citation: SCIENCE CHINA Information Sciences **53**, 1738 (2010); doi: 10.1007/s11432-010-4056-z

View online: https://engine.scichina.com/doi/10.1007/s11432-010-4056-z

View Table of Contents: https://engine.scichina.com/publisher/scp/journal/SCIS/53/9

Published by the Science China Press

SCIENCE CHINA
Information Sciences

• **RESEARCH PAPERS** •

September 2010 Vol. 53 No. 9: 1738–1752
doi: 10.1007/s11432-010-4056-z

Precise bounded-concurrent zero-knowledge proofs for NP

DING Ning[*] & GU DaWu

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, *China*

Received April 22, 2009; accepted April 27, 2010

Abstract Precise concurrent zero-knowledge is a new notion introduced by Pandey et al. in Eurocrypt'08. This notion captures the idea that the view of any verifier in concurrent interaction can be reconstructed in almost the same time. Pandey et al. also constructed some precise concurrent zero-knowledge argument systems. In this paper we construct a precise bounded-concurrent zero-knowledge proof for NP, which has the precision $p(n, y) = \text{poly}(n) + O(ny)$. Bounded-concurrency means that an a-priori bound on the number of concurrent sessions is specified before the protocol is constructed. Our result holds even if adversarial verifiers adopt the dynamic scheduling strategy. We make no setup assumption. The advantage of proof systems over argument systems is that the soundness property of proof systems can resist computationally-unbounded adversarial provers, while that of argument systems can only resist polynomial-time adversarial provers.

Keywords interactive proofs and arguments, zero-knowledge, precise zero-knowledge, proofs of knowledge, bounded concurrency

Citation Ding N, Gu D W. Precise bounded-concurrent zero-knowledge proofs for NP. Sci China Inf Sci, 2010, 53: 1738–1752, doi: 10.1007/s11432-010-4056-z

1 Introduction

Zero-knowledge proofs were introduced by Goldwasser et al. [1]. 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. The refinement of [2] calls for a tighter coupling between the expected running-time of V and that of S: a proof is zero-knowledge with tightness $t(\cdot)$ if there exists a fixed polynomial $p(\cdot)$ such that the expected running-time of $S(x)$ is upper-bounded by $t(|x|)$ times the expected running-time of $V(x)$ plus $p(|x|)$.

Micali and Pass [3] argued, however, that such coupling may still be insufficient, even when the tightness function is a constant and the polynomial $p(\cdot)$ is identically 0. Consider a malicious verifier V that, on input an instance $x \in \{0, 1\}^n$, takes n^{10} computational steps with probability $\frac{1}{n}$, and n steps the rest of the time. The expected running-time of V is $\Omega(n^9)$, and thus zero-knowledge with optimal tightness only requires that V be simulated in expected time $\Omega(n^9)$. They thought that it is doubtful to take indifference for V to get out and interact with the prover or to stay home and run S for granted. Since by interacting

[*]Corresponding author (email: cs.dingning@gmail.com)

with P, V will almost always execute n steps of computation, while (in absence of extra guarantees) running the simulator might always cause him to invest n^9 steps of computation. This discussion shows that we need a stronger notion of zero-knowledge.

Hence ref. [3] put forward a stronger notion: precise zero-knowledge[1)]. This notion captures the idea that 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). Thus, precise zero-knowledge bounds the knowledge of the verifier in terms of its actual computation. Ref. [3] also showed that any language outside BPP cannot possess black-box zero-knowledge protocols with polynomial precisions (so all known concurrent black-box zero-knowledge protocols for NP are imprecise). Furthermore, ref. [4] showed that the simulator in [5] of Barak's non-black-box zero-knowledge arguments cannot provide polynomial precisions either.

However, a more realistic setting for zero-knowledge is the concurrent setting. A zero-knowledge protocol is concurrent zero-knowledge if for every polynomial-time verifier there exists a polynomial-time simulator that can output an indistinguishable view in concurrent execution of the protocol. So concurrent zero-knowledge is also formalized as a worst-case notion, which also suffers the same problems ref. [3] proposed. Hence a very recent work by Pandy et al. [6] proposed the notion of precise concurrent zero-knowledge. They constructed some precise concurrent zero-knowledge argument systems. Since argument systems only have computational soundness, i.e., the soundness holds against polynomial-time adversaries, we are interested in the question how to construct precise proof systems in concurrent setting.

Our result. We answer this question affirmatively with respect to bounded-concurrent setting. By bounded-concurrency we mean that an a-priori bound on the number of concurrent sessions is specified before the protocol is constructed. That is, we construct a precise bounded-concurrent zero-knowledge proof for NP. Our result holds even if the verifier adopts the dynamic scheduling strategy. We make no setup assumption. More formally, our result can be described as follows.

Theorem 1.1. Assuming the existence of $k(n)$-round perfectly-hiding commitments, there exist $\omega(kn \log n)$-round bounded-concurrent zero-knowledge proofs for NP with precision $p(n, y) = \text{poly}(n) + O(ny)$.

Our technique. Ref. [3] presented an approach to obtain polynomial/linear precisions (in stand-alone setting): Record the verifier's running-time in the first time and "cut off" the simulation whenever the verifier attempts to run for too long in the rewind. The significant characteristic of this technique is that the first run of the simulated interaction is used to generate the transcript and continue the interaction, while the rewind run is used to gather information (for extraction).

Our simulator adopts this "cut-off" technique and extends it to bounded-concurrent setting with more complicated probability analysis. So the simulator accesses the verifier in a non-black-box way. We stress that our simulator will not adopt the recursive strategy (presented in [6–8]) but use a straight-line fashion to simulate concurrent sessions.

Other related works. The notion of concurrent zero-knowledge [9] formalizes security in a scenario in which several verifiers access concurrently a prover and maliciously coordinate their actions so as to extract information from the prover. The first concurrent zero-knowledge argument system for NP has been given by [10] that showed that $O(n^{\varepsilon})$ are sufficient for any $\varepsilon > 0$. Poly-logarithmic round-complexity was achieved in [7] and, finally, ref. [8] showed that $\widetilde{\Omega}(\log n)$ rounds are sufficient. Ref. [11] showed that in the black-box model $\widetilde{\Omega}(\log n)$ rounds are necessary for concurrent zero knowledge for non-trivial languages.

The bounded-concurrent setting was first put forward explicitly in [5], which presented bounded-concurrent non-black-box zero-knowledge arguments which have many properties such as constant-round, public coins and polynomial-time simulation which cannot be achieved by any black-box zero-knowledge protocol for a non-trivial language simultaneously.

1) The original work [3] used the term "local zero-knowledge". Ref. [4] changed to use the term "precise zero-knowledge"

2 Preliminaries

We adopt the standard way of modeling an efficient adversary as a family of probabilistic polynomial-sized circuits. Computational indistinguishability refers to indistinguishability by non-uniform polynomial-sized adversaries. See ref. [2] for the definitions of the notions of negligible functions, computational and statistical indistinguishability. This paper will also use the following cryptographic primitives and for lack of space we refer readers to the related literature for their extensive definitions and known constructions. These primitives are commitment schemes [2], interactive proofs [1] and arguments [13], zero-knowledge [1, 14, 15], witness indistinguishability (WI) [16] and proofs of knowledge [1, 17–19].

Probabilistic notations. In this paper the whole sample space Ω underlying probabilistic analysis always refers to all outcomes of finite random coins of randomized algorithms. Let A, B be two events. We use AB (or $A \cdot B$, $A \cap B$) to denote the intersection of A and B, $A + B$ to denote the union of A and B if they are disjoint, $\overline{A}$ to denote the complement of A, $A - B$ to denote the difference of A and B, i.e., the set of points that belong to A but not belong to B. $\Pr[AB]$ (resp. $\Pr[A + B]$, $\Pr[\overline{A}]$) is the probability of AB (resp. $A + B$, $\overline{A}$), and $\Pr[A|B]$ is the conditional probability of A on the occurrence of B.

A decomposition of Ω (resp. A) is a set of $\{D_1, \ldots, D_d\}$, where d is a natural number, satisfying $D_i \cap D_j = \phi$ for any $1 \leqslant i \neq j \leqslant d$ and $\sum_{i=1}^{d} D_i = \Omega$ (resp. $\sum_{i=1}^{d} D_i = A$).

Definition 2.1 (Concurrent execution)**.** Let (P, V) be a two-party protocol, V^* be any interactive machine, and $\{(a_i, b_i)\}_{i=1}^{t}$ be a set of t inputs to the protocol (P, V). A t-time concurrent execution of (P, V) coordinated by V^* on inputs $\{(a_i, b_i)\}_{i=1}^{t}$ is the following experiment:

1. Run t independent copies of P with the ith copy getting a_i as input;
2. Provide V^* with the $b_1, \ldots, b_t$;
3. On each step V^* outputs a message (k, m). The kth copy of P is given with the message m. V^* is given with the prover's response.

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 t-time concurrent execution of (P, V), for any circuit (i.e., machine with auxiliary input) V^*, denote by $v = (x_1, \ldots, x_t, (m_1, m_2, \ldots, m_{ut}))$ the view of V^* coordinating t sessions. Then denote by $\mathrm{STEPS}_{V^*}(v)$ the number of computational steps taken by V^* running on input $x_1, \ldots, x_t$ and letting the jth message received be m_j. (In counting steps, we follow the assumption in [3] that an algorithm A, given the code of a second algorithm B and an input x, can emulate the computation of B on input x with linear-time overhead.)

Definition 2.2 (Precise concurrent zero-knowledge) [6]**.** Let (P, V) be an interactive proof or argument system for a language $L = L(R)$, and let $p : N \times N \to N$ be a monotonically increasing 2-variate function. We say that (P, V) is a concurrent zero-knowledge proof or argument with precision p if there exists a probabilistic algorithm S such that for every polynomial-sized V^* and every polynomial $g(n)$ and every list $\{(x_i, w_i)\}_{i=1}^{g(n)}$, $(x_i, w_i) \in R$:

1. The following two ensembles are computationally indistinguishable:

a) The view of V^* in a $g(n)$-time concurrent execution of (P, V) with inputs $\{(x_i, w_i), x_i\}_{i=1}^{g(n)}$.

b) $S(x_1, \ldots, x_{g(n)}, V^*)$.

2. For every sufficiently long $r \in \{0, 1\}^*$, let v be the view generated by S_r on input $(x_1, \ldots, x_{g(n)}, V^*)$. Then $\mathrm{STEPS}_{S_r(x_1, \ldots, x_{g(n)}, V^*)} \leqslant p(n, \mathrm{STEPS}_{V^*}(v))$.

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

Remark 2.3. We say (P, V) is bounded-concurrent zero-knowledge with precision p if the sessions number is restricted to n in Definition 2.2. Since the security parameter can be "scaled", this means that for every fixed polynomial $q(x)$, we can construct a protocol from (P, V) that remains zero-knowledge when executed $q(n)$ times concurrently. Furthermore, it can be seen that if the precision $p(n, y)$ of (P, V) is a polynomial in y and the degree on y is d, then the one of the "scaled" protocol is still a polynomial in y with degree d.

Remark 2.4. Since in general the running-time of S is required to be bounded by a polynomial, the case where $p(n, y)$ is a super-polynomial in n or y is trivial. Thus in this paper we say that (P, V) is precise (resp. imprecise) if S can (resp. cannot) provide a non-trivial precision p.

3 Our result

In this section we present the proof of Theorem 1.1. by constructing a precise bounded-concurrent zero-knowledge proof for NP. In section 3.1 we present the zero-knowledge protocol and show that it is an interactive proof for NP. In section 3.2 we give an overview of the precise simulator for this protocol and then present its actual description. In sections 3.3–3.5 we show that bounded-concurrent zero-knowledge with the required precision can be achieved via this simulator and thus complete the proof of Theorem 1.1.

3.1 The zero-knowledge proof

In this subsection we present the construction of the zero-knowledge protocol. We first introduce the cryptographic ingredients underlying the protocol and then present its actual description. Further, we prove that this protocol is an interactive proof for NP.

Protocol 3.1. The zero-knowledge proof for NP.

Public input: $x \in \{0,1\}^n$ (statement to be proved is "$x \in L$")	$w \quad x$
	$\downarrow \quad \downarrow$
Prover's auxiliary input: w (a witness that $x \in L$)	$\boxed{P} \quad \boxed{V}$
Stage 1	
Step P1.0. Prover sends first message for perfectly-hiding commitment.	$\xrightarrow{msg}$
Step V1.1. Verifier chooses a string $\sigma \leftarrow_{\mathrm{R}} \{0,1\}^n$ and $s \in \{0,1\}^{\mathrm{poly}(n)}$.	
Send the commitment *com* to σ to prover using randomness s.	$\xleftarrow{com}$
Then verifier proves to prover in the m sequential composition of the perfectly WI AOK $(P,V)_{\mathrm{atom}}$	
that there exist σ, s such that $com(\sigma, s) = com$. i.e., for $i = 1, \ldots, m$ do the following.	
Step P1.i.1. Prover sends first message for perfectly-hiding commitment.	$\xrightarrow{msg_i}$
Step V1.i.2. Verifier sends com_i of the ith $(P,V)_{\mathrm{atom}}$ to prover.	$\xleftarrow{com_i}$
Step P1.i.3. Prover sends a random challenge ch_i to verifier.	$\xrightarrow{ch_i}$
Step V1.i.4. Verifier completes the ith $(P,V)_{\mathrm{atom}}$ by sending $resp_i$.	$\xleftarrow{resp_i}$
Stage 2	
Steps P,V2.X: Prover proves to verifier using its input w in n parallel of Blum's 3-round	
proof for HC, where V uses σ as the challenge, that $x \in L$:	
Step P2.1. Send n first messages of the Blum's proof system.	$\xrightarrow{com_{HC}}$
Step V2.2. Reveal the commitment *com*, i.e., send σ and s.	$\xleftarrow{\sigma, s}$
Step P2.3. Compute the n answers to the challenge σ and send the answers.	$\xrightarrow{resp_{HC}}$

3.1.1 *Cryptographic ingredients*

Perfectly-hiding commitments. The first ingredient underlying our protocol is perfectly-hiding commitments. Any construction presented in [12] and [2] (sec. 4.8.2.3) is suitable for our protocol. For simplicity we describe our protocol with respect to the 2-round construction from claw-free functions [2] in which the receiver runs the 1st round by sending a message for the commitment and then in the 2nd round the committer sends its commitment to the receiver. We will use *msg* and *com* to denote the two messages of the scheme respectively. Sometimes we abuse notations a little by also using *com* to represent the commitment algorithm.

Perfectly WI AOKs. The second ingredient is perfectly WI AOKs for NP relations. Instantiated with a (2-round) perfectly-hiding commitment scheme, the n parallel executions of Blum's protocol is a perfectly WI AOK for NP relations. The knowledge extractor of this protocol on two different valid answers with respect to a common commitment can output the witness (if the answers are consist of the committed message). In this paper we will adopt the m sequential composition of the WI system for some m. We will use $msg_i, com_i, ch_i, resp_i$ to denote the four messages of the ith atomic protocol for $1 \leqslant i \leqslant m$.

3.1.2 *Actual description*

Let $x \in \{0,1\}^n$ be the public input to be proven in an NP language L. Our protocol consists of two stages. In the first stage the verifier first chooses a random string $\sigma \in \{0,1\}^n$ and sends the commitment to σ to the prover using the perfectly-hiding commitment scheme. It can be seen all commitments and their decommitments form an NP relation. Denote the perfectly WI AOK for this relation by $(P,V)_{\rm atom}$. Choose an arbitrary m satisfying $m = \omega(n \log n)$. Then following this commitment the verifier proves to the prover that it knows the decommitment via an m sequential repetitions of $(P,V)_{\rm atom}$, which is still a perfectly WI AOK. In the second stage the prover proves to the verifier that $x \in L$ in a slightly modified Blum's proof in which the verifier opens the commitment to σ and uses σ as the challenge. The actual description of the protocol is described as Protocol 3.1 shows.

First we should guarantee that for Protocol 3.1 the completeness and soundness hold, as the following theorem states.

Theorem 3.2. Protocol 3.1 is an interactive proof for NP.

Proof. We show that the completeness and soundness properties are satisfied.

Completeness: Straightforward. If $(x,w) \in R_L$, the truthful prover can use w for $x \in L$ as the witness in Stage 2 to make the verifier convinced.

Soundness: The proof idea of this part is as follows. Since the whole interaction of Stage 1 can be viewed as a big perfectly-hiding commitment, we deduce that when reaching Stage 2 the prover has no idea about the value of challenge σ. In other words, even though the cheating prover reaches the second stage after seeing all messages in the first stage, the messages in the second stage are independent of the verifier's messages in Stage 1. So a cheating prover violating the soundness of our protocol can be transformed to a (full power) cheating prover violating the soundness of (the n parallel executions of) Blum's proof. We omit the details of the proof of this part due to lack of space.

Thus, to prove Theorem 1.1, we only need to prove the following theorem.

Theorem 3.3. Protocol 3.1 is bounded-concurrent zero-knowledge with precision $p(n,y) = \mathrm{poly}(n) + O(ny)$.

To prove Theorem 3.3, we need to construct a simulator and analyze its running-time and output. In section 3.2 we present a construction of the precise simulator. In sections 3.3–3.5 we show that the simulator has all required properties and thus complete the proof of Theorem 3.3.

3.2 The precise simulator

In this subsection we construct a simulator for Protocol 3.1 which will be proven to be able of providing bounded-concurrent zero-knowledge with the required precision with respect to verifier's dynamic scheduling.

3.2.1 *Overview*

Recall that Protocol 3.1 has an m-time repetition of $(P,V)_{\rm atom}$ in Stage 1 that is special-sound. Hence the main idea underlying our simulation strategy is to rewind the verifier to execute some $(P,V)_{\rm atom}$s twice in simulating every session. For a specific session on receiving two different convincing transcripts (with respect to a common commitment), the simulator calls the extractor of $(P,V)_{\rm atom}$ to compute the verifier's challenge σ of this session. After learning σ, the simulator can complete the interaction in Stage 2 of this session. Moreover, the simulator adopts the "cut-off" technique presented by [3] in every rewind to obtain the precision.

However, a simple strategy of rewinding the verifier in a session will lead the information gathered in other sessions to become useless since the verifier may choose different challenges in Step V1.1 of other sessions if these verifier's steps are in the relevant part of the rewind. So the simulator cannot use the extracted σ's to take part in the corresponding Blum's proofs. Hence the simulator has to re-extract all this information, which results in that the simulator runs in super-polynomial time. This problem, previously encountered in constructing concurrent zero-knowledge, has been successively solved by [7, 8](without ensuring precision) and [6] with oblivious recursive simulation strategies. But the analysis of the simulator's running-time and output is quite complicated.

Our simulator will not follow the recursive strategy but to use a straight-line fashion, which will make the following security analysis simpler. Like the one in [3], our simulator uses the first run of the relevant part in a rewind to generate the view, while the rewind run of this part is used to gather information for extraction. Further, we would like to use the term "the ordinary interaction/ordinary run" instead of the first run (of a relevant part) in the following of this paper. Messages generated in rewind run are only for extraction and will be abandoned then.

Notations. Throughout this section we will use k, t to index a session (i.e., $1 \leqslant k, t \leqslant n$), and use superscript j to index an overall prover/verifier message (i.e., $1 \leqslant j \leqslant u \cdot n$, u is the verifier's round number of Protocol 3.1). We will use subscript p and v to denote that the message is a prover's or verifier's message. We will use parenthesized superscript to denote the session to which a message belongs (e.g. $msg_i^{(k)}$, $com_i^{(k)}$, $ch_i^{(k)}$, $resp_i^{(k)}$). We will sometimes drop the session number when it is clear from the context (e.g. ch_i, $resp_i$). Further, we will drop the subscript i if we do not need to refer to i explicitly (e.g. ch, $resp$). We will call $[m_p^{j_0}, m_v^{j_1}]$ a rewind interval or interval if the pair $(m_p^{j_0}, m_v^{j_1})$ is $(ch, resp)$ of some $(P, V)_{\text{atom}}$ of some session in the ordinary interaction. We will say a rewind occurs on the interval $[ch, resp]$ if on receiving $resp$ the simulator goes back to the step that it sent ch and re-interact with the verifier in this relevant part. So we will use ch' and $resp'$ (if there is) to denote messages in the rewind run corresponding to ch and $resp$. We will use $com_{HC}, resp_{HC}$ to denote the prover's messages for Step P2.1 and P2.3 of n parallel composition of slightly modified Blum's proof of Protocol 3.1.

Our simulator uses the honest prover's strategy to interact with the verifier. Without lost of generality, we omit describing Step P1.0 of n sessions in simulation by assuming the simulator can send these messages (for Step P1.0) to the verifier in the same schedule as n real provers do. So we consider the number of prover's messages as nu. We also assume that every real prover will respond to verifier's message immediately while the verifier interleaves n sessions in a way it pleases. Hence we only need to describe the simulator's behavior on receiving different verifier's messages.

Our simulator will behave as real provers until it receives a verifier's response of a $(P, V)_{\text{atom}}$ of a session. But it will not rewind the verifier on receiving every $resp_i$, $1 \leqslant i \leqslant m$, of every session. Note that there are m $(P, V)_{\text{atom}}$s for the simulator to interact with the verifier in a specific session. Since the sessions number is n, this means that for this specific session there are at least $m - n$ values for i such that when i equals any one of these $m - n$ values, there is no verifier's message for Step V2.2 of other sessions contained in the interval $[ch_i, resp_i]$. We basically let the simulator rewind the verifier in extracting the verifier's challenge of this session on receiving $resp_i$ where i is one of these values. (Note that in the rewind the verifier may use the dynamic scheduling. But the simulator is oblivious of the dynamic scheduling and interacts with the verifier in this rewind to wait for $resp_i'$ unless it receives a verifier's message for Step V2.2 of some session.) More precisely, on receiving $resp_i$ where i is one of these $m - n$ values, the simulator chooses a random ch_i' and rewinds the verifier to re-execute the interaction of this rewind until the verifier outputs $resp_i'$. In the case that V^* sends a message for Step V2.2 of some session, the simulator terminates the rewind. Moreover, the simulator uses the "cut-off" technique, i.e., it records the running-time of the verifier in every round in the ordinary interaction and then uses this running-time of the verifier to bound the verifier's computing in the rewind run. That is, if the verifier cannot output $resp_i'$ after running this time bound, the simulator terminates this rewind run and continues the ordinary interaction (i.e., the interaction following $resp_i$).

When it succeeds in extracting the verifier's challenge in an interval of a session, the simulator will not perform rewinds in the following intervals of this session. If the verifier sends an invalid message in the ordinary interaction of a session, the simulator terminates the interaction of this session. When the simulation reaches Stage 2 of a session and it has not obtained the verifier's challenge, the simulator terminates the entire simulation.

Since in a rewind run, the verifier may not send a right response or it has not enough time to finish its computing or it outputs a message for Step V2.2 of some session or although its responses in the ordinary and rewind runs are valid, the verifier may open the commitment in one of the two runs in a value different from the committed value in the first message of some atomic protocol, the simulator would fail to extract verifier's challenge in this rewind with some probability. But we will show that the simulator fails only with a negligible probability after many rewinds for every session. Lastly, note that every verifier's message in the ordinary interaction is contained in at most n intervals of n sessions. So we will show that the verifier's running-time in all rewind runs is no more than that in the ordinary interaction times n, which accordingly ensures the required precision.

3.2.2 *Actual description*

Our simulator's operation follows the above description. We now turn to formally describe the simulator's algorithm:

Algorithm 3.4. The simulator S

Input:

$x_1,\dots,x_n \in \{0,1\}^n$: the statement to be proven in the ith session is that $x_i \in L$.

$V^* \in \{0,1\}^{\mathrm{poly}(n)}$: the description of a polynomial-sized verifier coordinating an n-time concurrent execution.

Initialization: The simulator constructs such tables and variables as follows.

1. G: a two-dimension table of length $n\times n$. $G[k][t]$ contains a boolean variable: $VMsgStepV2.2$. When the simulation goes to the step at which the simulator receives $resp$ (denote its corresponding prover's challenge by ch) of a $(P,V)_{\mathrm{atom}}$ of the kth session, $G[k][t].VMsgStepV2.2=1$ if the verifier's message for Step V2.2 of the tth session is in the interval $[ch, resp]$. Otherwise, $G[k][t].VMsgStepV2.2=0$. Initially, $G[k][t].VMsgStepV2.2=0$ for every $1\leqslant k,t\leqslant n$.

2. B: a table of length n. For every $1\leqslant k\leqslant n$, $B[k]$ contains two variables: $IsExtSuc$ and $ExtVal$. $B[k].IsExtSuc=0$ if the verifier's challenge σ of the kth session has not been extracted in the current step in the simulation. Otherwise, $B[k].IsExtSuc=1$. In this case, $B[k].ExtVal$ is σ. Initially, $B[k].IsExtSuc=0$ and $B[k].ExtVal=null$ for every $1\leqslant k\leqslant n$.

3. $RunTime$: a table of length $n\cdot u$. For every $1\leqslant j\leqslant n\cdot u$, $RunTime[j]$ records V^*'s running-time, on receiving the $(j-1)$th prover's message, in outputting the jth verifier's message in the ordinary interaction. Initially, $RunTime[j]=0$ for every $1\leqslant j\leqslant n\cdot u$.

Simulating each step: For $j=1,\dots,u\cdot n$ the simulator computes the jth prover's message m_p^j in the following way: Feed the previously computed messages $(m_p^1,\dots,m_p^{j-1})$ to V^* to obtain jth verifier's message (k, m_v^j) (where k is the session number this message belongs to). In the same time S counts the running-time of V^* in outputting the jth verifier's message and stores it into $RunTime[j]$. On receiving the jth verifier's message, S checks whether this message is valid (e.g., a valid decommitment). If it is not, S terminates the simulation of the kth session. Otherwise, compute the prover message m_p^j according to the current step in the simulated proof of the kth session:

Step P1.1.1: If the verifier's message is for Step V1.1 of the kth session, then S sends the first message of the perfectly hiding commitment scheme. The jth prover's message will be this message.

Step P1.*i*.3, $1\leqslant i\leqslant m$**:** If the verifier's message is for Step V1.i.2 of the kth session, for $t=1,\dots,n$, set $G[k][t].VMsgStepV2.2=0$. Then choose a random ch. The jth prover's message m_p^j will be ch.

Step P1.i.1, $1 < i \leqslant m$**:** If the verifier's message is for Step V1.$(i-1)$.4 of the kth session, then S does the following:

1. If one of the following conditions holds, go to the next step: 1) There exists a t satisfying $G[k][t].VMsgStepV2.2 = 1$; 2) $B[k].IsExtSuc = 1$. Otherwise, S starts to rewind V^*. That is, denote this verifier's message by $resp$, and its corresponding prover's challenge by ch. Assume ch is the j_0th prover's message in the ordinary interaction. Then S chooses a random ch'. Replace $m_p^{j_0}$ by ch', feed $(m_p^1, \ldots, ch')$ to V^* and interact with V^* in the rewind until V^* outputs $resp'$. In this rewind run, S follows the honest prover strategy without doing any further rewind. Moreover, S counts V^*'s running-time in this rewind run. If V^* sends a message for Step V2.2 of some session, or it outputs an invalid message, or it cannot complete the computing to output $resp'$ within time $\sum_{j'=j_0+1}^{j} RunTime[j']$, or $ch = ch'$, S terminates this rewind and moves to the next step. Otherwise, S calls the extractor of $(P, V)_{\mathrm{atom}}$ on two different accepting answers to obtain the verifier's challenge σ. If the extractor succeeds, set $B[k].IsExtSuc = 1$ and $B[k].ExtVal = \sigma$.
2. Send the first message of the perfectly hiding commitment scheme. The jth prover's message will be this message.

Step P2.1: If the verifier's message is for Step V1.m.4 of the kth session, then S computes the first prover's message com_{HC} of the Blum's proof using the extracted secret. Concretely, if $B[k].IsExtSuc = 0$, this means that S failed to extract the verifier's challenge. Then S terminates the simulation. Otherwise, S computes com_{HC} by employing Blum's simulation strategy with the knowledge of the verifier's challenge stored in $B[k].ExtVal$. The jth prover's message will be com_{HC}.

Step P2.3: If the verifier's message is for Step V2.2 of the kth session, then S does the following:

1. If the σ in this verifier's message is different from that one S extracted, then S terminates the simulation.
2. For $t = 1, \ldots, n$, $t \neq k$, set $G[t][k].VMsgStepV2.2 = 1$.
3. Compute $resp_{HC}$ of the Blum's proof. The jth prover's message will be $resp_{\mathrm{HC}}$.

Thus the actual description of the simulator is fully depicted. To complete the proof of Theorem 3.3, we need to show the following facts.

1. As S differs from n provers in the real interaction in that S does not have the witnesses for $x_1, \ldots, x_n$, it can finish the interaction only if it can extract the verifier's challenges of n sessions before the interaction entries Stage 2 of these sessions. Hence we need to ensure that S can extract all verifier's challenges of n sessions with an overwhelming probability.
2. S's output is computationally indistinguishable from the view of V^* in a real interaction.
3. S's running-time is bounded by $n+1$ times V^*'s running-time on the view generated by S (plus a polynomial).

In the following subsections we will prove these facts.

3.3 The simulator's success probability

In this subsection we show that the simulator can extract the verifier's challenges of n sessions with an overwhelming probability even if the verifier adopts the dynamic scheduling, as the following claim states.

Claim 3.5. S can extract all verifier's challenges of n sessions with an overwhelming probability (with respect to the verifier's dynamic scheduling).

Proof. Let us order the n sessions in an arbitrary way. Assume that V^* is deterministic since it is a polynomial-sized circuit, and S first chooses sufficiently long coins for both the ordinary interaction and all possible rewinds (the sample space can be considered as the products of the space of coins for ordinary interaction and the spaces of coins for all possible rewinds, and without lost of generality assume that outcomes of each space are of same length) and then starts simulation. Each outcome of the (finite) coins corresponds to an instantiation of the interaction. In the following we first present some conventions, state the proof idea and then present the details of the proof.

Conventions. We make some conventions as follows.

1. In an instantiation of the interaction if the interaction reaches an interval $[ch, resp]$ and this interval does not contain any V^*'s message for Step 2.2 of some session, we say that the simulator can perform the rewind on this interval. Moreover, in the rewind run of this interval V^* still outputs a valid answer $resp'$ (within the time bound), we say that the simulator *almost succeeds* in extracting V^*'s challenge of this session. Furthermore, if in the two runs S's challenges ch and ch' are different and both of V^*'s two responses are the very committed value in the first message of this atomic protocol (together with some auxiliary de-commitment information), then S can extract V^*'s challenge and we say that in this case S succeeds in this interval.

2. In an instantiation of the interaction if S succeeds in extracting V^*'s challenge in an interval of a session, then according to the simulator algorithm S will not perform any rewind for this session. But we still say that S succeeds in all residual intervals of this session.

3. In an instantiation of the interaction if V^* outputs an invalid answer in the ordinary interaction of some session, S will terminate the session. In this case S actually succeeds in simulating this session since the real prover will behave identically. If this V^*'s message occurs in Stage 1, we still say that S succeeds in those intervals of this session containing or posterior to this V^*'s invalid message.

Proof idea. The proof idea can be depicted as follows. We first show that S can succeed in every session with an overwhelming probability and then show it succeeds in all sessions still with an overwhelming probability. It can be seen that a key task in the proof is to show that S succeeds in each session with an overwhelming probability. To do this, we consider a variant of S that is oblivious of the "cut-off" technique. Denote this variant simulator by S', which differs from S in that S' does not adopt the "cut-off" technique in every rewind. Further, in any rewind if V^*'s running-time in outputting a valid response in the rewind run is more than the time bound, S' does not run the extractor to compute σ. (The previous conventions are also applied to S'.) Thus, S' succeeds in an interval of a session if and only if S succeeds in this interval.

Actually, S' and S are two almost identical algorithmic processes on the same probability space, and some concerned events involving the latter can be captured by the events involving the former. Hence in this proof we first present some analysis on some events, probabilities involving the former. When we achieve these useful results, we will use them to obtain the desired results on the events and probabilities involving the latter (in a little implicit manner).

Events. To state the probability analysis, we define the following events.

1. Let $CanRwd_{ti}$, $1 \leqslant i \leqslant m$ and $1 \leqslant t \leqslant n$, denote the event that S' can perform the rewind in the ith interval of the tth session.

2. Let $AlmSuc_{ti}$, $1 \leqslant i \leqslant m$ and $1 \leqslant t \leqslant n$, denote the event that S' can perform the rewind in the ith interval of the tth session and V^* outputs a valid response in the rewind run. So it can be seen that $AlmSuc_{ti} \subset CanRwd_{ti}$.

3. Let $SAlmSuc_{ti}$, $1 \leqslant i \leqslant m$ and $1 \leqslant t \leqslant n$, denote the event that S can perform the rewind in the ith interval of the tth session and V^* outputs a valid response (within the time bound) in the rewind run of the ith interval of the tth session. So $SAlmSuc_{ti}$ is actually the event that S almost succeeds in the ith interval of the tth session and it is a subset of $AlmSuc_{ti}$.

4. Let $Succ_{ti}$, $1 \leqslant i \leqslant m$ and $1 \leqslant t \leqslant n$, denote the sub-event of $SAlmSuc_{ti}$ which additionally satisfies that S's challenges in the ordinary run and rewind run are different and V^*'s responses in the two runs are the very value committed to in the first message of the atomic protocol together with some auxiliary de-commitment information. It is easy to see that $\Pr[SAlmSuc_{ti}]-\Pr[Succ_{ti}] = neg(n)$ ($neg(n)$ denotes an unspecified negligible function) due to the computational binding property of the underlying commitment scheme and the fact that the challenges are identical only with negligible probability.

5. Let $Success_t$, $1 \leqslant t \leqslant n$, denote the event that S succeeds in the tth session. Let $Success$ denote the event that S succeeds in all sessions.

We stress that the definitions of the above events obey the previous conventions. Namely, assuming S' (resp. S) succeeds in the i_0th interval of session t, or V^* sends an invalid message in the i_0th interval of session t in the ordinary interaction, we view that $CanRwd_{ti}$, $AlmSuc_{ti}$ (resp. $SAlmSuc_{ti}$ and $Succ_{ti}$)

occur for all $i \geqslant i_0$.

In the language of events, this claim says $\Pr[Success] = 1 - neg(n)$. To prove this we need to show $\Pr[Success_t] = 1 - neg(n)$ for all t's, which is actually the main body of this proof. To this end we use the induction method to argue that for session t, there exist $\omega(\log n)$ good intervals such that the probability S fails in a good interval on the occurrence that S fails in all previous good intervals is bounded by a constant. (Note that to make the conditional probability meaningful the probability of the conditional event should be more than 0 strictly. If the conditional event, i.e., S fails in all previous good intervals, occurs with probability 0, this means S has succeeded. That is, $\Pr[Success_t] = 1$, which is of course a nice case. Thus we only need to present the analysis of the case in which the conditional event occurs with probability more than 0.) Thus S fails in all the $\omega(\log n)$ good intervals only with negligible probability.

We illustrate the proof with respect to the first good interval in detail and then show the existence of the residual good intervals inductively. That is, we show the existence of the first good interval and S succeeds in this interval with at least a constant probability (Claim 3.5.1–Claim 3.5.3). Then we show the existence of the residual good intervals and S succeeds in each of these intervals on the occurrence that S fails in the intervals prior to the one also with at least a constant probability, and thus S succeeds in session t with an overwhelming probability (Claim 3.5.4).

Claim 3.5.1. For each t, there exist at least $\omega(n \log n)$ values for i satisfying the requirement of $\Pr[CanRwd_{ti}] \geqslant a$, where a is a constant and $0 < a < 1$. In particular, denote by i_1 the minimal one among these $\omega(n \log n)$ values and then $\Pr[CanRwd_{ti_1}] \geqslant a$. (Here i_1 depends on t.)

Proof. Since the sessions number is n (i.e., there are n V^*'s messages for Step V2.2), for every session there are at least $m - n$ intervals on which S' can perform the rewind in any instantiation. So we have $\Pr[CanRwd_{t1}] + \Pr[CanRwd_{t2}] + \cdots + \Pr[CanRwd_{tm}] \geqslant m - n$. We show this fact as follows. Suppose N is the number of all instantiations and β_i, $1 \leqslant i \leqslant m$, is the number that S' can perform the rewind in the ith interval of session t in all instantiations. Thus, $\Pr[CanRwd_{ti}] = \frac{\beta_i}{N}$. So $\beta_1 + \cdots + \beta_m$ is the number that S' can perform the rewinds of session t in all instantiations. On the other hand, S' can perform at least $m - n$ rewinds for session t in every instantiation and thus it can perform at least $(m - n)N$ rewinds for session t in all instantiations. This means $\beta_1 + \cdots + \beta_m \geqslant (m - n)N$. Divide both sides by N. The fact follows.

Since $m = \omega(n \log n)$, we can choose functions c, f and a constant $c_0 > 1$ satisfying $c = \omega(\log n)$, $f = \omega(\log n)$ and $m = cn > f(c_0 n + 1) > n$ (for sufficiently large n's). We claim there are at least $(c - c_0)n$ numbers in $\{\Pr[CanRwd_{ti}] : 1 \leqslant i \leqslant m\}$ satisfying that these numbers are equal to or more than $1 - \frac{1}{c_0}$. This fact is shown as follows. Suppose there are q numbers in $\{\Pr[CanRwd_{ti}] : 1 \leqslant i \leqslant m\}$ which are less than $1 - \frac{1}{c_0}$. Then $\Pr[CanRwd_{t1}] + \Pr[CanRwd_{t2}] + \cdots + \Pr[CanRwd_{tm}] < q(1 - \frac{1}{c_0}) + (m - q)$. So $q(1 - \frac{1}{c_0}) + (m - q) > m - n$. Then $q < nc_0$. This means that there are at least $m - q > (c - c_0)n$ numbers satisfying they are equal to or more than $1 - \frac{1}{c_0}$. Denote the subscripts of the least $(c - c_0)n$ numbers of them by $a_1^{(1)}, a_2^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}$, listed in an increasing order. Let i_1 denote $a_1^{(1)}$ and a denote $1 - \frac{1}{c_0}$. Then $\Pr[CanRwd_{ti_1}] \geqslant a$.

Claim 3.5.2. Assume t, i_1 and a are the ones appearing in Claim 3.5.1. Then $\Pr[AlmSuc_{ti_1}] > \frac{a^3}{8}$.

Proof. Note that a rewind in an interval only re-executes the interaction on and posterior to the first message of this interval, and the random coins used in the interaction (including the ordinary interaction and all rewinds) prior to this interval are not involved in the rewind run of this interval. Thus we decompose the whole sample space according to the coins used prior to the i_1th interval of session t into $\{D_1, D_2, \ldots, D_d\}$, where the outcomes of coins in D_j, $1 \leqslant j \leqslant d$, share common coins used in the interaction prior to this interval (those outcomes resulting in that S' has succeeded in session t before reaching this interval constitute a class, without lost of generality denoted D_d), and d is the number of the decomposition. Then $AlmSuc_{ti_1} = D_1 AlmSuc_{ti_1} + D_2 AlmSuc_{ti_1} + \cdots + D_d AlmSuc_{ti_1}$. The motivation of introducing the decomposition is to reduce evaluating $\Pr[AlmSuc_{ti_1}]$ to evaluating the probabilities of $AlmSuc_{ti_1} D_j$'s.

Let us first evaluate $\Pr[AlmSuc_{ti_1} D_j]$. To do this we further decompose $CanRwd_{ti_1} D_j$ into $\{E_{j1}, \ldots,$

$E_{jd_j}\}$ for some d_j, where the outcomes in E_{jk}, $1 \leqslant k \leqslant d_j$, further share common coins used in the ordinary run of this interval (we regard $CanRwd_{ti_1}D_d$ as the decomposition of itself). Since S' chooses independent coins in the ordinary run and rewind run of the interval, it can be seen that on the occurrence of E_{jk} the probability that S' receives V^*'s valid message in the rewind (i.e., $AlmSuc$ occurs) equals the probability that S' can perform the rewind (i.e, $CanRwd_{ti_1}$ occurs) on the occurrence of D_j. Namely, $\Pr[AlmSuc_{ti_1}|E_{jk}] = \Pr[CanRwd_{ti_1}|D_j]$ for each k. (We stress that this equality holds no matter whether S' has succeeded in session t or not before reaching interval i_1th due to our conventions.) Then

$$\begin{aligned}
\Pr[AlmSuc_{ti_1}D_j] &= \Pr[CanRwd_{ti_1} \cdot AlmSuc_{ti_1}D_j] \\
&= \Pr[AlmSuc_{ti_1}(E_{j1} + \cdots + E_{jd_j})] \\
&= \sum_{k=1}^{d_j} \Pr[AlmSuc_{ti_1}E_{jk}] \\
&= \sum_{k=1}^{d_j} \Pr[E_{jk}]\Pr[AlmSuc_{ti_1}|E_{jk}] \\
&= \sum_{k=1}^{d_j} \Pr[E_{jk}] \cdot \Pr[CanRwd_{ti_1}|D_j] \\
&= \Pr[CanRwd_{ti_1}D_j] \cdot \Pr[CanRwd_{ti_1}|D_j] \\
&= \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j]^2.
\end{aligned}$$

Hence $\Pr[AlmSuc_{ti_1}] = \sum_{j=1}^{d} \Pr[AlmSuc_{ti_1}D_j] = \sum_{j=1}^{d} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j]^2$.

Let us do some preparation before evaluating the lower bound for $[\Pr[AlmSuc_{ti_1}]$. Denote by $I = \{j_1, j_2, \ldots, j_k\}$ the subset of $\{1, \ldots, d\}$ such that $\Pr[CanRwd_{ti_1}|D_j] > \frac{a}{2}$ for all $j \in I$. (I can be empty.) Hence for $j \notin I$, $\Pr[CanRwd_{ti_1}|D_j] \leqslant \frac{a}{2}$. Let b denote $\sum_{j\in I} \Pr[D_j]$. (If I is empty, then let $b = 0$.) Then $\sum_{j\notin I} \Pr[D_j] = 1 - b$.

As $CanRwd_{ti_1} = \sum_{j=1}^{d} CanRwd_{ti_1}D_j$ and $\Pr[CanRwd_{ti_1}] = \sum_{j=1}^{d} \Pr[CanRwd_{ti_1}D_j]$, it follows from Claim 3.5.1 that

$$\begin{aligned}
a \leqslant \Pr[CanRwd_{ti_1}] &= \sum_{j=1}^{d} \Pr[CanRwd_{ti_1}D_j] \\
&= \sum_{j\in I} \Pr[CanRwd_{ti_1}D_j] + \sum_{j\notin I} \Pr[CanRwd_{ti_1}D_j] \\
&= \sum_{j\in I} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j] + \sum_{j\notin I} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j] \\
&< \sum_{j\in I} \Pr[D_j] + \sum_{j\notin I} \frac{a}{2} \cdot \Pr[D_j] \\
&= b + (1-b) \cdot \frac{a}{2}.
\end{aligned}$$

So $b + (1-b) \cdot \frac{a}{2} > a$ and then $b > \frac{a}{2-a} > \frac{a}{2}$. (Note that we obtain a lower bound for b that is independent of the decomposition.) Now the lower bound for $\Pr[AlmSuc_{ti_1}]$ can be evaluated as follows.

$$\begin{aligned}
\Pr[AlmSuc_{ti_1}] &= \sum_{j=1}^{d} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j]^2 \\
&= \sum_{j\in I} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j]^2 \\
&\quad + \sum_{j\notin I} \Pr[D_j] \cdot \Pr[CanRwd_{ti_1}|D_j]^2
\end{aligned}$$

$$\begin{aligned}&> \sum_{j\in I}\frac{a^2}{4}\cdot\Pr[D_j]\\&= \frac{a^2}{4}\cdot\sum_{j\in I}\Pr[D_j]=\frac{a^2b}{4}>\frac{a^3}{8}.\end{aligned}$$

Claim 3.5.3. Still assume t, i_1 and a are the ones appearing in Claim 3.5.2. Then $\Pr[SAlmSuc_{ti_1}] > \frac{a^3}{16}$ and thus $\Pr[Succ_{ti_1}] > \frac{a^3}{16} - neg(n)$. (This shows interval i_1 is the 1st good one we find for session t.)

Proof. In the case of $AlmSuc_{ti_1} = SAlmSuc_{ti_1}$, the claim holds of course. Thus we only need to consider the case of $AlmSuc_{ti_1} \neq SAlmSuc_{ti_1}$ below. We define a mapping from $AlmSuc_{ti_1} - SAlmSuc_{ti_1}$ to $SAlmSuc_{ti_1}$. Let u be an outcome in $AlmSuc_{ti_1} - SAlmSuc_{ti_1}$. We define its image u' under this mapping as follows: As we mentioned, S chose sufficiently long coins for the ordinary run and all rewind runs. Denote by $coins_1$ the coins in u actually used for the ordinary run of this interval (the i_1th interval of the tth session), by $coins_2$ the coins in u actually used for the rewind run of this interval. Since V^* uses dynamic scheduling, $coins_1$ and $coins_2$ may be not of same length. Denote by l the length of the longer one between $coins_1$ and $coins_2$. Without lost of generality assume $|coins_1| = l$ and $|coins_2| \leqslant l$ (the mapping is similar for the opposite case). Denote by $coins_2'$ the coins consisting of $coins_2$ and some successive coins in the coins chosen for the rewind run of this interval satisfying $|coins_2'| = l$. Then exchange $coins_1$ and $coins_2'$ in u and retain other coins unchanged. Denote by u' the result. We show $u' \in SAlmSuc_{ti_1}$.

In fact, it follows from $u \in AlmSuc_{ti_1} - SAlmSuc_{ti_1}$ that when using u as coins S' can perform the rewind of this interval and V^* outputs a valid answer in the rewind and V^*'s running-time in the ordinary run is less than that in the rewind run. This means that there does not exist any verifier's message for Step V2.2 contained in both the ordinary run and rewind run. Hence we conclude that when using u' as coins S'/S can still perform the rewind on this interval and V^* outputs a valid answer in the rewind and V^*'s running-time in the ordinary run is more than that in the rewind run. Accordingly, $u' \in SAlmSuc_{ti_1}$.

Moreover, assume $u_1, u_2 \in AlmSuc_{ti_1} - SAlmSuc_{ti_1}$ are two different outcomes and u_1', u_2' are their images in $SAlmSuc_{ti_1}$ under this mapping. It can be seen that $u_1' \neq u_2'$. This means the mapping is injective and thus $|AlmSuc_{ti_1} - SAlmSuc_{ti_1}| \leqslant |SAlmSuc_{ti_1}|$. Hence $\Pr[SAlmSuc_{ti_1}] \geqslant \frac{1}{2}\Pr[AlmSuc_{ti_1}] > \frac{a^3}{16}$. Further, by the computational binding property of the underlying commitment scheme and the fact the two challenges of S'/S in the ordinary run and rewind run are identical only with negligible probability, $\Pr[Succ_{ti_1}] > \frac{a^3}{16} - neg(n)$.

Claim 3.5.4. For each t, $\Pr[Success_t] = 1 - neg(n)$.

Proof. For session t, Claims 3.5.1–3.5.3 show interval i_1 is a good one satisfying that $\Pr[Succ_{ti_1}] > \frac{a^3}{16} - neg(n)$. (In this proof we will continue to use the variables c, f and c_0, introduced in the proof of Claim 3.5.1.) We claim we can continue to find the next $f-1$ good intervals $i_2, i_3, \ldots, i_f$ satisfying $i_1 < i_2 < \cdots < i_f$ and $\Pr[Succ_{ti_1} \cup Succ_{ti_2} \cup \cdots \cup Succ_{ti_f}] \geqslant 1 - neg(n)$. Since $Success_t \supset Succ_{ti_1} \cup Succ_{ti_2} \cup \cdots \cup Succ_{ti_f}$, if the probability of latter is overwhelming then the claim follows.

Recall that in the proof of Claim 3.5.1, we showed that there are $(c - c_0)n$ numbers $a_1^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}$ in $[1, m]$ satisfying $\Pr[CanRwd_{ti}] \geqslant a$ for $i \in \{a_1^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}\}$ and we denoted $a_1^{(1)}$ by i_1. Then on the occurrence of $\overline{Succ}_{ti_1}$, there are still $m - n$ intervals on which S' can perform the rewind for session t in each instantiation. Even if S' can perform the rewind in all intervals except intervals $a_2^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}$, there are at least $(c - c_0)n - 1 - n$ ones among intervals $a_2^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}$ on which S' can perform rewind in any instantiation. Thus, we have

$$\begin{aligned}&\Pr[CanRwd_{ta_2}|\overline{Succ}_{ti_1}] + \Pr[CanRwd_{ta_3}|\overline{Succ}_{ti_1}] + \cdots\\&\quad + \Pr[CanRwd_{ta_{(c-c_0)n}}|\overline{Succ}_{ti_1}] \geqslant (c - c_0)n - 1 - n.\end{aligned}$$

Applying the same analysis in the proof of Claim 3.5.1, we have that there are at least $(c - 2c_0)n - 1$

numbers $a_1^{(2)} < a_2^{(2)} < \cdots < a_{(c-2c_0)n-1}^{(2)}$ in $\{a_2^{(1)}, \ldots, a_{(c-c_0)n}^{(1)}\}$ satisfying $\Pr[CanRwd_{ti}|\overline{Succ}_{ti_1}] \geqslant a$ for $i \in \{a_1^{(2)}, a_2^{(2)}, \ldots, a_{(c-2c_0)n-1}^{(2)}\}$. Let i_2 denote $a_1^{(2)}$.

Let us consider the probability space induced by $\overline{Succ}_{ti_1}$. That is, the induced sample space is $\overline{Succ}_{ti_1}$ and the induced probability is denoted by $\Pr_{\overline{Succ}_{ti_1}}[\cdot]$, which is defined as $\Pr_{\overline{Succ}_{ti_1}}[A] = \Pr[A|\overline{Succ}_{ti_1}]$ for every event A.

Applying the same probability analysis in the proofs of Claim 3.5.2 and Claim 3.5.3 into the induced probability space, we have $\Pr_{\overline{Succ}_{ti_1}}[Succ_{ti_2}] > \frac{a^3}{16} - neg(n)$. That is, $\Pr[Succ_{ti_2}|\overline{Succ}_{ti_1}] > \frac{a^3}{16} - neg(n)$ and thus $\Pr[\overline{Succ}_{ti_2}|\overline{Succ}_{ti_1}] \leqslant 1 - \frac{a^3}{16} + neg(n)$. (Interval i_2 is the second good one we find.)

Inductively (we omit the details since the proof is essentially the same as previously), on the occurrence of $\overline{Succ}_{ti_1} \cdots \overline{Succ}_{ti_{k-1}}$ where $1 < k \leqslant f$, there are $(c - kc_0)n - k + 1$ numbers $a_1^{(k)} < a_2^{(k)} < \cdots < a_{(c-kc_0)n-k+1}^{(k)}$ in $\{a_2^{(k-1)}, \ldots, a_{(c-kc_0)n-k+1}^{(k-1)}\}$ satisfying $\Pr[CanRwd_{ti}|\overline{Succ}_{ti_1} \cdots \overline{Succ}_{ti_{k-1}}] \geqslant a$ for $i \in \{a_1^{(k)}, a_2^{(k)}, \cdots, a_{(c-kc_0)n-k+1}^{(k)}\}$. (Note that $(c - fc_0)n - f + 1 > 0$.) Denote $a_1^{(k)}$ by i_k. We have $\Pr[Succ_{ti_k}|\overline{Succ}_{ti_1} \cdots \overline{Succ}_{ti_{k-1}}] > \frac{a^3}{16} - neg(n)$ and $\Pr[\overline{Succ}_{ti_k}|\overline{Succ}_{ti_1} \cdots \overline{Succ}_{ti_{k-1}}] \leqslant 1 - \frac{a^3}{16} + neg(n)$. (Intervals i_k's are all good intervals we find.) Hence,

$$\begin{aligned}
\Pr[Success_t] \geqslant\ & \Pr[Succ_{ti_1} \cup Succ_{ti_2} \cup \cdots \cup Succ_{ti_f}] \\
&= 1 - \Pr[\overline{Succ}_{ti_1}\overline{Succ}_{ti_2} \cdots \overline{Succ}_{ti_k}] \\
&= 1 - \Pr[\overline{Succ}_{ti_1}] \cdot \Pr[\overline{Succ}_{ti_2}|\overline{Succ}_{ti_1}] \cdots \\
&\quad \Pr[\overline{Succ}_{ti_k}|\overline{Succ}_{ti_1} \cdots \overline{Succ}_{ti_{k-1}}] \\
&\geqslant 1 - \left(1 - \frac{a^3}{16} + neg(n)\right)^f \\
&= 1 - neg(n).
\end{aligned}$$

The claim follows.

So far we have shown that $\Pr[Success_t] = 1 - neg(n)$ for each t. Thus,

$$\begin{aligned}
\Pr[Success] =\ & \Pr[Success_1 \cap \cdots \cap Success_n] \\
&= 1 - \Pr[\overline{Success}_1 \cup \cdots \cup \overline{Success}_n] \\
&\geqslant 1 - \sum_{t=1}^{n} \Pr[\overline{Success}_t] \\
&= 1 - n \cdot neg(n) = 1 - neg(n).
\end{aligned}$$

We complete the proof of Claim 3.5.

3.4 The simulator's output distribution

Claim 3.5 ensures that S can obtain the verifier's challenges of the n sessions with very high probability. Then we need to show that S's output is computationally indistinguishable from the view of V^* in a real interaction, as the following claim states.

Claim 3.6. For any sequence $\{(x_t, w_t)\}_{t=1}^n$ such that w_i is the witness for $x_i \in L$, the following two random variables are computationally indistinguishable:

1. The view of V^* in an n-time concurrent execution of Protocol 3.1 with inputs $\{(x_t, w_t), x_t\}_{t=1}^n$. We denote this variable by X.
2. The output of S on input $(x_1, \ldots, x_n, V^*)$. We denote this variable by Y.

Proof. This claim can be proved by using the hybrid argument. Due to short of space, we only sketch the proof. We construct a hybrid simulator $\widehat{S}$ which on input $(\{(x_t, w_t)\}_{t=1}^n, V^*)$ follows the same strategy as S in Stage 1 of all sessions. In simulating steps of Stage 2 of the tth session it checks whether it has succeeded in extracting V^*'s challenge. If it has not succeeded, $\widehat{S}$ terminates the simulation as S does. Otherwise, $\widehat{S}$ will provide w_t as input to the honest prover algorithm to output the messages of Stage 2.

Let Z denote the output of $\widehat{S}$. Then we can show that Y and Z are computationally indistinguishable and that X and Z are statistically indistinguishable. Thus the claim follows.

3.5 The simulator's running time

Thus all that is left to prove is the following claim, which shows the precision can be achieved.

Claim 3.7. For every sufficiently long $r \in \{0,1\}^*$, let $v = S_r(x_1, \ldots, x_n, V^*)$. Then there exists a monotonically increasing 2-variate function $p(n,y) = \text{poly}(n) + O(ny)$ such that $\text{STEPS}_{S_r(x_1,\cdots,x_n,V^*)} \leqslant p(n, \text{STEPS}_{V^*}(v))$.

Proof. Choose any sufficiently long coins r for S. On input $(x_1, \ldots, x_n, V^*)$ S outputs the simulated view v. $RunTime[j]$ stores the running-time of V^* on receiving m_p^{j-1} in outputting m_{v}^j. $\text{STEPS}_{V^*}(v)$ is $\sum_{j=1}^{un} RunTime[j]$.

According to Algorithm 3.4, an interval of a session includes some verifier's messages of different sessions. In the instantiation of the interaction corresponding to r, assume there are p intervals of n sessions which can be rewound and the qth interval includes verifier's messages $\{m_v^{j_q}, \ldots, m_v^{j'_q}\}$, where $1 \leqslant q \leqslant p$. Letting T_q denote V^*'s running-time in the rewind run of interval q, we have $T_q \leqslant RunTime[j_q] + \cdots + RunTime[j'_q]$ for all q's.

Moreover, it can be seen that every verifier's message in the ordinary interaction is contained in at most n intervals of n sessions. This means that for every j, there are at most n values for q satisfying that $RunTime[j]$ appears in the above representation of the upper bound for T_q. So it follows that $T_1 + \cdots + T_p$ is less than $n \cdot \sum_{j=1}^{un} RunTime[j]$. Namely, V^*'s total running-time in all rewinds is less than $n \cdot \sum_{j=1}^{un} RunTime[j]$.

Hence the entire simulation time consists of three parts: (1) S's running-time adopting the honest prover strategy to interact with V^* in both ordinary interaction and all rewind runs. (2) V^*'s running-time in both ordinary interaction and all rewind runs. (3) the time for extraction. First, we know the time described in (1) is no more than $(n+1)$ times the honest prover's running-time and thus it is a polynomial. Second, it can be seen that it takes S polynomial time to carry out judging operations and reading and writing tables and extracting verifier's challenges, etc. So the time for (3) is also a polynomial. Denote the sum of the time described in (1)(3) by $p_1(n)$. Third, the time described in (2) is less than $(n+1) \cdot \sum_{j=1}^{un} RunTime[j]$.

Considering the linear-time overhead of emulating machines, $\text{STEPS}_{S_r}(x_1, \ldots, x_n, V^*)$ is less than $p_1(n) + l(n+1) \cdot \text{STEPS}_{V^*}(v))$, where l is the linear-time overhead. Clearly, there is a monotonically increasing polynomial $p(n,y) = \text{poly}(n) + O(ny)$ such that $\text{STEPS}_{S_r}(x_1, \ldots, x_n, V^*) \leqslant p(n, \text{STEPS}_{V^*}(v))$. This completes the proof.

Thus, we complete the proof of Theorem 3.3. Combining it with Theorem 3.2, we also finish the proof of Theorem 1.1.

4 Conclusions

This paper is focused on the question how to construct precise concurrent zero-knowledge proofs for NP. As a result, we propose a bounded-concurrent zero-knowledge proof for NP with precision $p(n,y) = \text{poly}(n) + O(ny)$. The previously known precise concurrent zero-knowledge systems are arguments. Hence the soundness property of our result holds with respect to computationally-unbounded adversaries, while that of arguments holds only with respect to polynomial-time adversaries.

Acknowledgements

This work was supported by the Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 200802480019).

References

1 Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof-systems. In: Proc 17th STOC. New York: ACM, 1985. 291–304

2 Goldreich O. Foundations of Cryptography – Basic Tools. Cambridge: Cambridge University Press, 2001

3 Micali S, Pass R. Local zero knowledge. In: Proc 38th STOC. New York: ACM, 2006. 306–315

4 Pass R. A precise computational approach to knowledge. Dissertation for the Doctoral Degree. Cambridge: MIT, 2006

5 Barak B. How to go beyond the black-box simulation barrier. In: Proc 42nd FOCS. Los Alamitos: IEEE, 2001. 106–115

6 Pandey O, Pass R, Sahai A, et al. Precise concurrent zero-knowledge. In: Proc Eurocrypt'08. LNCS 4965. Berlin: Springer-Verlag, 2008. 397–414

7 Kilian J, Petrank E. Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: Proc STOC'01. New York: ACM, 2001. 560–569

8 Prabhakaran M, Rosen A, Sahai A. Concurrent zero knowledge with logarithmic round-complexity. In: Proc 43rd FOCS. Los Alamitos: IEEE, 2002. 366–375

9 Dwork C, Naor M, Sahai A. Concurrent zero knowledge. In: Proc 30th STOC. New York: ACM, 1998. 409–418

10 Richardson R, Kilian J. On the concurrent composition of zero-knowledge proofs. In: Proc Eurocrypt'99. LNCS 1592. Heidelberg: Springer-Verlag, 1999. 415–431

11 Canetti R, Kilian J, Petrank E, et al. Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: Proc 33rd STOC. New York: ACM, 2001. 570–579

12 Damgard I, Pedersen T, Pfitzmann B. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In: Proc Crypto'93. LNCS 773. Heidelberg: Springer-Verlag, 1993. 250–265

13 Brassard G, Chaum D, Crépeau C. Minimum disclosure proofs of knowledge. J Comput Syst Sci, 1988, 37: 156–189

14 Blum M. How to prove a theorem so no one else can claim it. In: Proc of the International Congress of Mathematicians, Berkeley, California, USA, 1986. 1444–1451

15 Goldreich O, Micali S, Wigderson A. Proofs that yields nothing but their valid or all languages in NP have zero-knowledge proof systems. J ACM, 1991, 38: 169–192

16 Feige U, Shamir A. Witness indistinguishability and witness hiding protocols. In: Proc 22nd STOC. New York: ACM, 1990. 416–426

17 Feige U, Fiat A, Shamir A. Zero-knowledge proofs of identity. J Crypt, 1988, 1: 77–94

18 Bellare M, Goldreich O. On defining proofs of knowledge. In: Proc Crypro'92, LNCS 740. Berlin: Springer-Verlag, 1992. 390–420

19 Tompa M, Woll H. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In: Proc 28th FOCS, Los Alamitos: IEEE, 1987. 472–482