Impossibility and Feasibility Results for Zero Knowledge with Public Keys[⋆]

Joël Alwen[1], Giuseppe Persiano[2], and Ivan Visconti[2]

[1] Technical University of Vienna, A-1010 Vienna, Austria
e9926980@stud3.tuwien.ac.at
[2] Dipartimento di Informatica ed Appl., Università di Salerno,
84081 Baronissi (SA), Italy
{giuper, visconti}@dia.unisa.it

Abstract. In this paper, we continue the study of the round complexity of black-box zero knowledge in the bare public-key (BPK, for short) model previously started by Micali and Reyzin in [11]. Specifically we show the impossibility of 3-round concurrent (and thus resettable) black-box zero-knowledge argument systems with sequential soundness for non-trivial languages. In light of the previous state-of-the-art, our result completes the analysis of the round complexity of black-box zero knowledge in the BPK model with respect to the notions of soundness and black-box zero knowledge.

Further we give sufficient conditions for the existence of a 3-round resettable zero-knowledge *proof* (in contrast to argument) system with concurrent soundness for $\mathcal{NP}$ in the upperbounded public-key model introduced in [14].

1 Introduction

The classical notion of a zero-knowledge proof system has been introduced in [1]. Roughly speaking, in a zero-knowledge proof system a prover can prove to a verifier the validity of a statement without releasing any additional information. Recently, starting with the work of Dwork, Naor, and Sahai [2], the concurrent and asynchronous execution of zero-knowledge protocols has been considered. In this setting, several concurrent executions of the same protocol take place and a malicious adversary controls the verifiers and the scheduling of the messages.

Motivated by considerations regarding smart cards, the notion of *resettable zero knowledge* (rZK, for short) was introduced in [3]. An rZK proof remains "secure" even if the verifier is allowed to tamper with the prover and to reset the prover in the middle of a proof to any previous state, then asking different questions. It is easy to see that concurrent zero knowledge is a special case of resettable zero knowledge and, currently, rZK is the strongest notion of

[⋆] The work presented in this paper has been supported in part by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT.

V. Shoup (Ed.): Crypto 2005, LNCS 3621, pp. 135–151, 2005.

zero knowledge that has been studied when security against malicious verifiers is considered. Unfortunately, in the plain model, if we only consider black-box zero knowledge, constant-round concurrent (and therefore resettable) zero knowledge is possible only for trivial languages (see [4]). Moreover, the existence of a constant-round non-black-box concurrent zero-knowledge argument system is currently an open question (see [5] for the main results on non-black-box zero knowledge). An almost constant-round non-black-box concurrent zero-knowledge argument system has been recently given in [6] by assuming the existence of only one (stateful) prover.

Such negative results have motivated the introduction of the *bare public-key model* [3] (BPK, for short). Here each possible verifier deposits a public key pk in a public file and keeps private the associated secret information sk. From then on, all provers interacting with such a verifier will use pk and the verifier can not change pk from proof to proof. Note that in the BPK model there is no interactive preprocessing stage, no trusted third party or reference string and the public file can be completely under the control of the adversary. Consequently the BPK model is considered a very weak set-up assumption compared to some previously proposed models [2,7,8,9,10]. In this model, however, the notion of soundness is more subtle. This was first noted in [11], where the existence of four distinct and increasingly stronger notions of soundness: one-time, sequential, concurrent and resettable soundness is shown. Moreover it was pointed out that the constant-round rZK argument system in the BPK model of [3] did not seem to be *concurrently sound.*

In [11], a 3-round one-time sound black-box rZK argument system and a 4-round sequentially sound black-box rZK argument system for $\mathcal{NP}$ in the BPK model are given. Moreover it is shown that in the BPK model neither zero knowledge in less than 3 rounds nor black-box zero-knowledge with resettable soundness are possible for non-trivial languages. Two main problems were left open in [11] (see page 553 of [11] and page 13 of [12]).

The first open problem, namely the existence of a constant-round rZK argument system with concurrent soundness in the BPK model has been recently solved in [13] where an (optimal) 4-round protocol is presented. Before this result, a 3-round resettable zero-knowledge argument system with concurrent soundness has been presented by Micali and Reyzin in [14] in the upperbounded public-key (UPK, for short) model, where the verifier has a counter and his public key can be used only an a-priori fixed polynomial number of times.

The second open problem, namely the existence of a 3-round {resettable, concurrent, sequential} zero-knowledge argument with sequential soundness in the BPK, has been very partially addressed in [13] where a 3-round sequentially sound sequential zero-knowledge argument system has been presented.

The most interesting open problem in the BPK model is therefore the existence of a 3-round resettable (or even only concurrent) zero-knowledge argument system with sequential soundness for $\mathcal{NP}$.

Our Contribution. In this paper we show that 3-round black-box concurrent (and therefore resettable) zero-knowledge argument systems with sequential soundness in the BPK model exist only for trivial languages. As a consequence, we have that in the BPK model, sequential soundness in 3 rounds can be achieved for non-trivial languages only when no more than black-box sequential zero knowledge is required. This is in contrast to both one-time soundness and concurrent soundness where 3 rounds and 4 rounds respectively, have been shown to be both necessary and sufficient for sequential, concurrent and resettable zero knowledge. Our result closes the analysis on the round complexity of the BPK model with respect to notions of soundness and zero knowledge, see Fig. 1.

The intuition behind our impossibility result goes as follows. In the impossibility proof of 3-round black-box zero knowledge given in [15] by Goldreich and Krawczyk (on which our result is based) a deciding machine runs the simulator to determine if x is in L. In case the simulator outputs an accepting transcript even when $x \notin L$, the work of the simulator can be used by an adversarial prover that violates the soundness of the protocol. The translation in [11] of this proof to the BPK model works when the proof system is sound against *concurrent* malicious provers (because a rewind can be simulated with a new concurrent session). The proof in this paper observes that for 3-round black-box concurrent zero knowledge, there is at least one proof that the simulator does *not* rewind and therefore sequential soundness will suffice.

We also give some sufficient conditions for achieving 3-round resettable zero-knowledge proof (in contrast to argument) systems with concurrent soundness in the UPK model. Moreover, our construction does not use assumptions with respect to superpolynomial-time algorithms (i.e., complexity leveraging).

	3-Round OTS	**3-Round SS**	**4-Round CS**
sZK	[MR Crypto 01]	[DPV Crypto 04]	Folklore
cZK	[MR Crypto 01]	Impossible [This Paper]	[DPV Crypto 04]
rZK	[MR Crypto 01]	Impossible [This Paper]	[DPV Crypto 04]

Fig. 1. The round complexity of black-box zero knowledge in the BPK model

2 The Public-Key Models

Here we describe the BPK and the UPK models that we consider for our results. We give the definitions of zero-knowledge proof and argument systems with respect to the notions of soundness and zero knowledge that we use in the paper. For further details, see [11,14,12].

The BPK model assumes that: 1) there exists a polynomial-size collection of records associating identities with public keys in the public file F; 2) an (honest) prover is an interactive deterministic polynomial-time algorithm that takes as input a security parameter 1^k, F, an n-bit string x, such that $x \in L$ where L is an $\mathcal{NP}$-language, an auxiliary input y, a reference to an entry of F and a random tape; 3) an (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in the following two stages: 1) in a first stage on input a security parameter 1^k and a random tape, V generates a key pair $(\mathtt{pk}, \mathtt{sk})$ and stores it's identity associated with $\mathtt{pk}$ in an entry of the file F; 2) in the second stage, V takes as input $\mathtt{sk}$, a statement $x \in L$ and a random string, V performs an interactive protocol with a prover, and outputs "accept" or "reject"; 4) the first interaction of a prover and a verifier starts after all verifiers have completed their first stage.

Definition 1. *Given an $\mathcal{NP}$-language L and its corresponding relation R_L, we say that a pair of probabilistic polynomial-time algorithms $\langle P, V\rangle$ is* **complete** *for L, if for all n-bit strings $x \in L$ and any witness y such that $(x, y) \in R_L$, the probability that V on input x when interacting with P on input x and y, outputs "reject" is negligible in n.*

Malicious Provers and Attacks in the BPK *Model.* Let s be a positive polynomial and $P^\star$ be a probabilistic polynomial-time algorithm that takes as first input 1^n.

$P^\star$ is an *s-sequential malicious prover* if it runs in at most $s(n)$ stages in the following way: in stage 1, $P^\star$ receives a public key $\mathtt{pk}$ and outputs an n-bit string x_1. In every even stage, $P^\star$ starts from the final configuration of the previous stage, sends and receives messages of a single interactive protocol on input $\mathtt{pk}$ and can decide to abort the stage in any moment and to start the next one. In every odd stage $i > 1$, $P^\star$ starts from the final configuration of the previous stage and outputs an n-bit string x_i.

P^* is an *s-concurrent malicious prover* if on input a public key $\mathtt{pk}$ of V, it can perform the following $s(n)$ interactive protocols with V: 1) if P^* is already running i protocols $0 \leq i < s(n)$ it can start a new protocol with V choosing a new statement to be proved; 2) it can output a message for any running protocol, receive immediately the response from V and continue.

Given an s-sequential malicious prover $P^\star$ and an honest verifier V, a *sequential attack* is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair $(\mathtt{pk}, \mathtt{sk})$ is obtained; 2) the first stage of $P^\star$ is run on input 1^n and $\mathtt{pk}$ and x_1 is obtained; 3) for $1 \leq i \leq s(n)/2$ the $2i$-th stage of $P^\star$ is run letting it interact with V which receives as input $\mathtt{sk}, x_i$ and a random string r_i, while the $(2i+1)$-th stage of $P^\star$ is run to obtain x_i.

Given an s-concurrent malicious prover P^* and an honest verifier V, a *concurrent attack* is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair $(\mathtt{pk}, \mathtt{sk})$ is obtained; 2) P^* is run on input 1^n and $\mathtt{pk}$; 3) whenever P^* starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and $\mathtt{sk}$.

Definition 2. *Given a complete pair $\langle P, V\rangle$ for an $\mathcal{NP}$-language L in the* BPK *model, then $\langle P, V\rangle$ is a* **concurrently** *(resp.,* **sequentially***)* **sound interactive argument system for** *L if for all positive polynomials s, for all s-concurrent (resp., s-sequential) malicious provers $P^\star$ and for any false statement "$x \in L$" the probability that in an execution of a concurrent (resp. sequential) attack, V outputs "accept" for such a statement is negligible in n.*

In the definition above, if the malicious prover $P^\star$ is computationally unbounded, then $\langle P, V\rangle$ is a proof (and not only an argument) system.

Definition 3. *Let $\langle P, V\rangle$ be an interactive proof or argument system for a language L. We say that a probabilistic polynomial-time adversarial verifier $V^\star$ is a* **concurrent adversary in the** BPK **model** *if on input polynomially many values $\bar{x} = x_1, \ldots, x_{\mathtt{POLY}(n)}$, it first generates the public file F with $\mathtt{POLY}(n)$ public keys and then concurrently interacts with $\mathtt{POLY}(n)$ number of independent copies of P (each with a valid witness for the statement), with common input $\bar{x}$ and without any restrictions over the scheduling of the messages in the different interactions with P. Moreover we say that the transcript of such a concurrent interaction consists of $\bar{x}$ and the sequence of prover and verifier messages exchanged during the interaction. We refer to $\mathtt{view}^P_{V^\star}(\bar{x})$ as the random variable describing the content of the random tape of $V^\star$ and the transcript of the concurrent interactions between P and $V^\star$.*

Definition 4. *Let $\langle P, V\rangle$ be an interactive argument or proof system for a language L in the* BPK *model. We say that $\langle P, V\rangle$ is black-box* **concurrent zero knowledge** *if there exists a probabilistic polynomial-time algorithm S such that for each polynomial-time concurrent adversary $V^\star$, let $S_{V^\star}(\bar{x})$ be the output of S on input $\bar{x}$ and black-box access to $V^\star$, then if $x_1, \ldots, x_{\mathtt{POLY}(n)} \in L$, the ensembles $\{\mathtt{view}^P_{V^\star}(\bar{x})\}$ and $\{S_{V^\star}(\bar{x})\}$ are computationally indistinguishable.*

Definition 5. *An interactive argument system $\langle P, V\rangle$ in the BPK model is black-box* **resettable zero knowledge** *if there exists a probabilistic polynomial-time algorithm S such that for all probabilistic polynomial time adversaries V^*, for all pairs of polynomials (s, t) and for all $x_i \in L$ where $|x_i| = n$ and $i = 1, \ldots, s(n)$, $V^\star$ runs in at most $t(n)$ steps and the following two distributions are indistinguishable:*

1. *the output of $V^\star$ that generates F with $s(n)$ entries and interacts (even concurrently) a polynomial number of times with each $P(x_i, y_i, j, r_k, F)$ where y_i is a witness for $x_i \in L$, $|x_i| = n$, j is the index (i.e. associated identity) of the public key of $V^\star$ in F and r_k is a random tape for $1 \le i, j, k \le s(n)$;*
2. *the output of S given black-box access to V^* on input $x_1, \ldots, x_{s(n)}$.*

*Moreover we define such an adversarial verifier V^* as an (s, t)***-resetting malicious verifier**.

The definitions in the UPK model are very similar to the ones given for the BPK model. The only interesting difference is the attack of the malicious prover. Indeed, in the UPK model there is a bound on the number of sessions that the malicious prover can open in order to prove a false statement. Indeed, the verifier only uses his public key an a-priori fixed polynomial number of times and he uses a counter (a value that is persistent between the sessions) to verify that his key is not yet expired. For details see [12].

3 3-Round Sequentially Sound cZK in the BPK Model

In this section we will concentrate entirely on the proof of the following theorem.

Theorem 1. *Any black-box concurrent zero-knowledge argument system satisfying sequential soundness in the* BPK *model for a language L outside of $\mathcal{BPP}$ requires at least* 4 *rounds.*

3.1 Techniques for Achieving the Result

In [11,12], it has been proven that in the BPK model concurrent soundness can not be achieved in less than 4 rounds. That proof mainly follows the proof of the following theorem by Goldreich and Krawczyk in [15]: "In the plain model, any black-box zero-knowledge argument system for a language outside of $\mathcal{BPP}$ requires at least 4 rounds". Indeed, the proof given in [15], crucially uses the fact that if there exists a simulator M, then the work of M can be used either to decide the language or to violate the soundness of the protocol. Indeed, in case the simulator outputs an accepting transcript for a false statement, an adversarial prover can prove the same false statement by emulating the rewinds of the simulator by means of concurrent sessions. Also in [11,12], the same analysis is carefully repeated by using a concurrent malicious prover, thus proving that black-box zero knowledge with concurrent soundness needs at least 4 rounds for non-trivial languages in the BPK model.

However, when the prover can only open sequential sessions, the previously discussed approach does not work anymore. Actually, only sequential soundness and zero knowledge are not enough to prove the impossibility result. Indeed, in [13], it has been shown that in the BPK model a 3-round sequential black-box zero-knowledge argument system with sequential soundness exists. It is therefore vital to use the *concurrent* black-box zero-knowledge property along with sequential soundness in order to obtain the desired claim. Note that the black-box concurrent zero-knowledge property (i.e., the existence of a strong simulator that works in an hostile setting) has been previously used in [16,17,4] to show that black-box concurrent zero knowledge can not be achieved respectively in 4, 7 and finally in a constant number of rounds in the plain model. However these previous results do not help at all in the BPK model, since constant-round black-box concurrent (and even resettable) zero knowledge has been achieved in the BPK model. In particular only 3 rounds are necessary for one-time soundness [11] and 4 rounds for concurrent soundness [13].

Our proof of Theorem 1 therefore exploits a joined use of techniques for proving impossibility results for 3-round black-box zero knowledge as well as impossibility results for black-box concurrent zero knowledge in the plain model.

High-Level Overview. For the sake of simplifying the presentation, we now only consider conversation-based protocols, i.e., protocols where at the end of each proof the verifiers decides to accept or to not accept the proof without using private data. Therefore, in a conversation-based protocol, by simply looking at the transcript of the protocol it is possible to efficiently decide whether the verifier accepts or not.

We will show that for any language L with a 3-round sequentially sound black-box concurrent zero knowledge argument system $\Pi = \langle P, V \rangle$ in the $\mathtt{BPK}$ model , we can use the simulator M for concurrent zero knowledge as a black box to design an efficient deciding machine D for L.

More precisely, instead of only working with the honest verifier (in contrast to the proofs of [11,15]) we will first design a concurrent adversarial verifier $V^\star$ (as well as a useful variant $\tilde{V}^\star$) which D will let interact with M on input "$x \in L$" in order to decide $x \in L$. Specifically D runs M against $V^\star$ and decides the language based on whether M outputs an accepting transcript. To show the correctness of D we design a $V^\star$ which opens nested sessions (which we refer to as levels), each corresponding to a different public key. To be precise the behavior of $V^\star$ is the following:

1. Upon receiving the first message of a session at level i, $V^\star$ initiates a new session at level $i + 1$ by using the $(i + 1)$-th entry of the public file, until $i = \mathtt{POLY}(k)$ (for some fixed polynomial $\mathtt{POLY}()$) and only continues with the session at level i once level $i + 1$ has been successfully completed.
2. Before sending the second message in the 3-round protocol $V^\star$ uses a family of pseudorandom functions indexed by its random tape and evaluated upon $V^\star$'s entire view of the current interaction with M (including all levels) in order to generate a new value to be used as a source of randomness for computing its response message for a given session.

In order to keep things as simple as possible during the proof, we will also describe a variant of $V^\star$ called $\tilde{V}^\star$ which acts just as $V^\star$ except at level j (where the value of j is specified as input at $\tilde{V}^\star$'s startup). For this special level $\tilde{V}^\star$ outputs all messages received from the simulator (i.e., its complete view of the interaction at all levels with M thus far) to its output tape and responds with messages read on its input tape. In other words $\tilde{V}^\star$ acts as a proxy for an external algorithm at level j. We will use $\tilde{V}^\star$ to contradict the soundness of Π in case of failure of D with a polynomially related probability.

Next we will define two mutually exclusive categories of executions of a simulator such that any possible execution of a concurrent simulator M must fall into precisely one of the two categories. In the proof we will show that although D only works for one of the two categories, the other can be simply ignored as such executions would need exponential time (in the security parameter k) and therefore can happen only with negligible probability (since the expected running time of the simulator is polynomial in k).

D decides whether x is in the language as follows: if M outputs an accepting transcript, D will accept x otherwise D will reject x. Notice that if $x \in L$, then by the fact that M is a simulator for concurrent zero knowledge, it follows that D will accept x with overwhelming probability. The case $x \notin L$ is more complex. We will design a cheating sequential prover $P^\star$ which runs $\tilde{V}^\star$ against M, using $\tilde{V}^\star$ as a proxy for sessions with an honest verifier V. Then we show that for any $x \notin L$, the probability that $P^\star$ succeeds in cheating an honest verifier V is polynomially related to the probability of M outputting an accepting transcript for x. Here "convinces" refers to the prover convincing the verifier of the theorem $x \in L$ in some session and $\approx_{poly}$ stands for polynomially related. Thus by soundness, M will only successfully prove a false statement with a negligible probability. Therefore we can conclude that D is a deciding machine for L contradicting the assumption that $L \notin \mathcal{BPP}$.

Now we shall begin the detailed discussion by describing the aforementioned adversary $V^\star$.

3.2 The Adversarial Verifier $V^\star$

In general $V^\star$ acts exactly as the honest verifier V would except for a few special deviations.

Initialization Phase. Let $\mathtt{POLY}()$ be some fixed polynomial. $V^\star$ (honestly and independently) generates $p = \mathtt{POLY}(k)$ public-private key pairs $\{(\mathtt{pk}_i, \mathtt{sk}_i)\}_i$ for $i \in \{1, 2, .., p\}$ placing all public keys $\mathtt{pk} = \{\mathtt{pk}_i\}_{i \leq p}$ in the public file F. That is $V^\star$ simply runs V's initialization algorithm p times, each time with a new uniformly and independently chosen random string, publishing all (public) output.

Interactive Phase. $V^\star$ maintains an internal counter i which is initialized to 1. The counter is used to keep track of the current level, i.e. the index of the public key in F which is to be used for that level.

We denote with a triple (a_i, b_i, c_i) a 3*-round protocol played at level* i, where a_i denotes the first message of the prover to the verifier, b_i the second message of the verifier to the prover, and c_i denotes the last message of the prover to the verifier.

Upon receiving a_i for $i < p$, $V^\star$ initiates a new session in a concurrent fashion, at level $i + 1$ requesting a proof of the same statement "$x \in L$". Only once this new level has been successfully completed, does it continue execution at level i. (If $i = p$ then $V^\star$ continues execution of the current session without initiating any new levels.) See Fig. 2.

At level i, once $V^\star$ has received a_i but before initiating the next level (and thus before choosing and sending b_i), $V^\star$ sets the string $r'_i = f_{r_i}(view_{V^\star})$ where:

1. r_i is $V^\star$'s random tape for this session.
2. $view_{V^\star_i}$ is $V^\star$'s view of the entire interaction at all levels so far up to level i.
3. $f_{r_i} \in \{PRF_r\}_{r \in R}$ is the pseudorandom function with the index r_i where R is the set of all possible random tapes with which $V^\star$ can be initialized.

Now for the rest of this session, $V^\star$ uses r'_i as its random tape. Once the new random string has been defined, $V^\star$ continues as the honest verifier does until the end of the session.

The $\tilde{V}^\star$ Variation. For reasons of simplicity and overview we will now describe a related verifier algorithm $\tilde{V}^\star$. The only difference between $V^\star$ and $\tilde{V}^\star$ is that

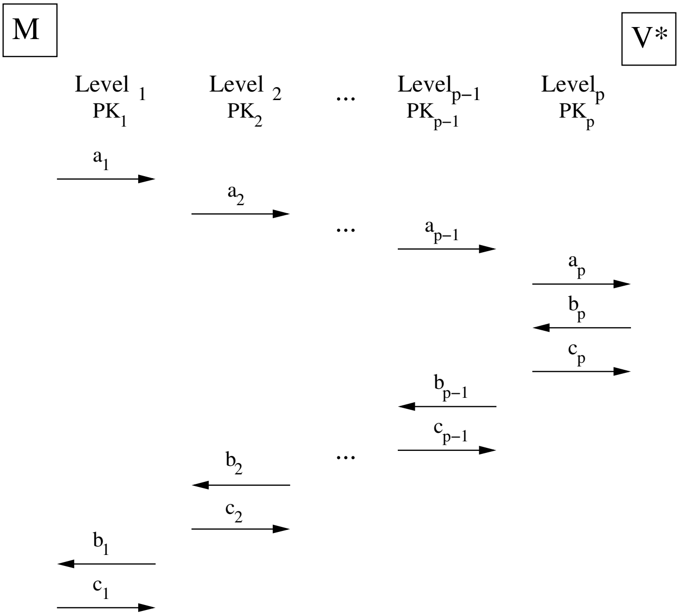

Fig. 2. The randomness used by $V^\star$ for each reply depends on his view

the latter, on initialization, reads an integer j along with a key $\mathtt{pk}$, on its input tape. $\tilde{V}^\star$ generates the public file just as $V^\star$ does, except for substituting $(V, \mathtt{pk})$ for the identity j. Further whenever $\tilde{V}^\star$ must send b_j, it writes its entire view $view_{\tilde{V}^\star_j}$ to its output tape[1], and pauses execution until a value for b_j is written to its input tape which it forwards to the prover as its message. Finally, upon receiving c_j, it writes c_j and the same view $view_{\tilde{V}^\star_j}$ it outputted when getting b_j at the beginning of this session. ($\tilde{V}^\star$ will later be run as a subroutine by another algorithm which will tell it which messages to use for level j but let it act just as $V^\star$ would at all other levels. $view_{\tilde{V}^\star_j}$ can be seen as an identifier for a given session.)

3.3 The Executions of the Simulator

Next we consider two (mutually exclusive) categories of executions of a simulator.

Definition 6. *Let M be the simulator that is guaranteed to exists for a black-box concurrent zero knowledge protocol $\Pi = \langle P, V, \rangle$ and let $V^\star$ be a probabilistic polynomial-time interactive algorithm. Then when M interacts with $V^\star$, if there exists a level j, $j \leq p$ with the following properties:*

[1] This view (which includes the random tape assigned by M) is the exact same view which $V^\star$ uses as the argument to its pseudorandom function when creating r'_j.

1. *Let $\{Views_{V_j^\star}\}$ be the set of all of views seen so far by $V^\star$ during the entire interaction when at level j. Then $view_{V_j^\star}$, the view of $V^\star$ just before playing b_j (at level j), is a new one, i.e. $view_{V_j^\star} \notin \{Views_{V_j^\star}\}$. Notice that the view at level j includes all messages a_i at levels i for $i \leq j$ (see Fig. 2).*
2. *From when b_j has been sent until when c_j has been sent, $V^\star$ never has a new view $view'_{V_j^\star} \notin \{Views_{V_j^\star}\}$. In other words after receiving b_j but before sending c_j for a given session at level j, M does not request a rewind such that $V^\star$ reaches a point where it sends a b'_j such that it has a view $view'_{V_j^\star}$ which it has never had before.*

then we call this execution of M a **j-deciding execution for $V^\star$***.*

Comments:

- If we consider a simulator M running against the adversarial verifier $V^\star$ specified above then the views considered in the definition are exactly those used as arguments to the pseudorandom function when setting r'_j.
- We use the term "j-deciding" because the simulator's j-th session will help to prove that it can be used to decide that language efficiently.

Definition 7. *Let M and $V^\star$ be as above. If there exists no j as above (that is, in order to complete every session resulting in an accepting view of $V^\star$, M requires at least one sequence of events (including a rewind) resulting in a new $View'_{V^\star}$ at level j) then we call this execution of M a* **hard execution for $V^\star$***.*

Comments and Examples. First we note that it is clear from the definition that given an execution of M, it must be either a j-deciding or hard for $V^\star$ but never both since either at least one such j exists or not. In order to clarify the two definitions we now give a simple example of a $V^\star$ with only one level of interaction with M.

Suppose $A = (a, b, c)$ and $B = (a', b', c')$ are the accepting transcripts of two different sessions (of the same protocol) and let R stand for "M rewinds $V^\star$". Now suppose M and $V^\star$ interact as follows:

$$(a, b, R, a', b', R, a, b, c) = (m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9)$$

In this case the triple (a, b, c) does not fulfill the requirement to make this execution of M a 1-deciding execution for $V^\star$ because although (m_1, m_2, m_9) is an accepting transcript, between m_2 and m_9 being received by $V^\star$, M rewinds $V^\star$ and causes it to have a new view (with messages m_3,m_4 and m_5) which violates the second point of the definition. Further although (m_7, m_8, m_9) is a transcript started and successfully completed without any rewind at all, this sequence violates the first point of the definition of a j-deciding execution since before playing m_8, $V^\star$ now has an old view: specifically that which it had before playing m_2. Therefore this is an hard execution for $V^\star$. If, on the other hand, the interaction began only with m_4, or if the last message were a message $m_{6'} = c'$ instead of m_6 then M would be a 1-deciding simulator for $V^\star$.

The intuition behind the crucial part of the proof is to show that if an execution of M when interacting with a specially designed adversarial verifier is a hard execution then M must perform an exponential number of rewinds in order to finish which implies that all hard executions with this verifier can only happen with negligible probability. If however they are j-deciding then in case the execution resulted in a false proof (i.e. if $x \notin L$) then the execution can be used by a specially designed malicious prover to break soundness. This is used to establish the correctness of a deciding machine which uses the simulator and an adversarial verifier as subroutines.

We now have all tools we need and can begin with the main proof.

3.4 The Proof of Theorem 1

Proof. Assume, by contradiction, there exists a language $L \notin \mathcal{BPP}$ with a 3-round black-box concurrent zero-knowledge argument system $\langle P, V\rangle$ in the BPK model enjoying sequential soundness, and let $V^\star$ and $\tilde{V}^\star$ be the adversarial verifiers as defined above. Then by the concurrent zero-knowledge property of $\langle P, V\rangle$, there exists an expected polynomial-time simulator M which, given oracle access to any concurrent verifier $V^\star$ will, for any true statement $x \in L$, output a transcript indistinguishable from that of $V^\star$ interacting with the honest prover P.

In order to reach the contradiction we construct an efficient deciding algorithm D which, on input x, decides membership in the language L which would imply $L \in \mathcal{BPP}$. To decide whether x is in L, D runs M while simulating $V^\star$ to it. If M outputs an accepting transcript, D will output $x \in L$, otherwise D will output $x \notin L$. Thus we will need to prove that with overwhelming probability M will output an accepting transcript if and only if $x \in L$.

Proposition 1. *If $x \in L$ then D will accept the proof with overwhelming probability.*

Proof. A session with a verifier using a pseudorandom tape looks the same as one with a verifier using a random tape, otherwise it is possible to break the randomness property of PRF. Since $V^\star$, after setting its random tape, uses the same algorithm as V for the rest of the interaction, $V^\star$ would, by completeness of $\langle P, V\rangle$, when interacting with P, accept for any given session at any level. Since there are only polynomially many sessions, but each has an overwhelming probability of resulting in an accepting view of $V^\star$, with overwhelming probability they will all result in an accepting view of $V^\star$. Thus with overwhelming probability $V^\star$ will accept when interacting with P. Therefore by the concurrent zero-knowledge property of $\langle P, V\rangle$, given a true statement as common input M will, with overwhelming probability, outputs an accepting view.

Proposition 2. *Given $x \notin L$ then D will reject x with overwhelming probability.*

Proof. To show this we will design a malicious prover $P^\star$ which can use M to prove any statement with a polynomially related probability to the probability of M outputting an accepting transcript for the same statement. We can then conclude that M outputs an accepting transcript for $x \notin L$ with negligible probability since otherwise the soundness of $\langle P, V\rangle$ is violated. However $P^\star$ will only work for j-deciding executions, so we will also show that an execution of M can be hard for $V^\star$ only with negligible probability.

We define $P^\star$ to be a polynomial-time algorithm with black-box access to M. Given x as input, $P^\star$ interacts with the honest verifier V with the goal of convincing V of the false statement "$x \in L$". It does this by running $\tilde{V}^\star$ with a random guess j' for j and the public key produced by V on its input, against M with the statement "$x \in L$". $P^\star$ then acts as a man-in-the-middle between $\tilde{V}^\star$ and V for all interaction at level j, and maintains a set of pairs: the views outputted by $\tilde{V}^\star$ and the corresponding response supplied by V. If $\tilde{V}^\star$ outputs a previously seen view then $P^\star$ does not forward the query to V, instead it simply answers by using the response previously stored in his memory. Further whenever $P^\star$ initiates a session with V (in order to get a response b_j for a session), $P^\star$ stores in a registry the view given in the output by $\tilde{V}^\star$. Thus when M outputs the message c_j of a session along with a view, $P^\star$ checks the value of the registry to see whether c_j corresponds to the currently open session with V. If this is the case then $P^\star$ forwards c_j to V thereby completing its current interaction with V. In other words the registry is used to store what is in essence, a unique ID for a session, namely the view which $V^\star$ would use as an argument when setting its pseudorandom tape r'. For a detailed description of $P^\star$ see Fig. 3.

Lemma 1. *When interacting with $P^\star$, M will act as when interacting with $V^\star$ with overwhelming probability.*

Proof. We must show that M can not tell the difference between interacting with $V^\star$ and with $P^\star$. Since $P^\star$ has oracle access to $\tilde{V}^\star$ it has complete control over its input including its random tape and can, in particular, perform all necessary rewinds and secret key operations. Thus it suffices to consider level j'. Because V and $V^\star$ follow the same algorithm in deciding the message b, apart from what they use as a random tape, the only concerns are rewinds and the fact that V's tapes are truly random and not chosen depending on any variables such as its view. Specifically:

1. M noticing that V is not being rewound when M requests a rewind to an old view.
2. M noticing that V is not being rewound when M requests a rewind to a new view (but in fact V simply uses a new random tape).
3. M noticing that V is not basing its randomness on the entire view at all levels.

The first concern is easily taken care of by pointing out in such a case $P^\star$ responds from memory, i.e. exactly as it did last time, as is expected, thus maintaining M's view. For the second concern, if M were able to tell the difference between $P^\star$ and $V^\star$ in such a situation, this would imply that it can tell the difference between a new random tape being used and a new pseudorandom tape. However such an M could then be used as a black box to create a distinguishing

Common input: security parameter k, public key $\mathtt{pk}$ and "$x \in L$".

$P^\star$ **Setup:**

1. $P^\star$ chooses a random $j' \leftarrow [1, p = \mathtt{POLY}(k)]$.
2. $P^\star$ initializes $\tilde{V}^\star$ with j' and $\mathtt{pk}$ on its input tape.
3. $P^\star$ initializes M with the public file generated by $\tilde{V}^\star$. Further $P^\star$ connects the communication tapes of M and $\tilde{V}^\star$ so that they interact with each other.
4. During the entire interaction $P^\star$ will maintain the set $S = \{View_{\tilde{V}^\star}, b_{j'}\}$ of views given in output by $\tilde{V}^\star$ at level j' and the corresponding responses written back by V. This set is initialized as empty.
5. During the entire interaction $P^\star$ will maintain the registry $View$ which is initialized as empty. (This will be used to keep track of the defining properties of the current open session with V.)
6. $P^\star$ initiates $\tilde{V}^\star$ with the statement "$x \in L$" all the while proxying between V and $\tilde{V}^\star$.

$P^\star$ **Interaction:**

- If $\tilde{V}^\star$ outputs a view $View_{\tilde{V}^\star}$ (we denote by $a_{j'}$ the most recent first round for level j' of $View_{\tilde{V}^\star}$) then $P^\star$ checks if the view is already in S:
    - TRUE: Respond with the corresponding $b_{j'}$ from S.
    - FALSE: Close any open session with V and begin a new one. Set the registry $View = View_{\tilde{V}^\star}$. Send $a_{j'}$ to V receiving $b_{j'}$ in response. Append $(View_{\tilde{V}^\star}, b_{j'})$ to S. And finally write $b_{j'}$ to $\tilde{V}^\star$'s input tape.
- If $\tilde{V}^\star$ outputs $(c_{j'}, View_{\tilde{V}^\star})$, then if $(a_{j'}, b_{j'}, c_{j'})$ is not an accepting transcript then $P^\star$ sends back an abort to $\tilde{V}^\star$; otherwise $P^\star$ checks if $View = View_{\tilde{V}^\star}$:
    - TRUE: Forward $c_{j'}$ to V thus completing the session with V.
    - FALSE: Drop $c_{j'}$.

Fig. 3. Using M as a black box to convince V of a false statement

algorithm for the PRF family which contradicts the families randomness property. This leaves the third concern which is dealt with by pointing out that a new session with V (where V will thus have a new random tape) resulting in a new response b is only requested by $P^\star$ upon receiving a new view. In other words the random tape used in generating the response b which M receives for a session, is created new only if the view at the beginning of a session is new. If the view is old then the old response (stored in memory by $P^\star$) is used, which implies (from M's point of view) that the same random tape was used to generate b. Thus the random tape used to produce the message received by M depends only and completely upon the view before playing b.

As we will show below $P^\star$ will succeed in its goal if and only if the execution of M is j-deciding for $V^\star$ and if it guesses the value of j correctly.

Lemma 2. *An execution of M is j-deciding for $V^\star$ with overwhelming probability.*

Proof. Suppose, by contradiction, that an execution of M is hard for $V^\star$. In this case we will show through analysis of the interaction between M and $V^\star$ that this execution of M needs exponential time. Specifically we show, by induction on the level i, that M must perform exponentially (in k) many rewinds.

By definition of a hard execution for $V^\star$, M must perform at least one rewind for each session it solves when interacting with $V^\star$. In particular this means that for any new session at level $i = p$, M must rewind. The same holds true at level $i-1$. Now since this is a hard simulator, for every level at least one of the rewinds leads to a new view upon receiving a_{i-1}. This implies that when $V^\star$ initiates a session at level i it will also have a new view, compared to all previous sessions at level i, and thus will use a new random string r'_i. In other words this will be a new session and will therefore require M to rewind again since this is a hard execution for $V^\star$.

Again since this is a hard execution for $V^\star$, there must be a rewind after b_{i-1} has been received which leads to a new view at level $i-1$, which means at least 2 new sessions at level i will need to be solved. Therefore at least 2 rewinds are required to solve all sessions at level i. The first rewind being used to solve the first session which must be successfully completed before b_{i-1} is sent, and the second rewind being used to solve the new session (with the new view) created after the rewind at level $i-1$.

Thus by induction M must perform at least: $\sum_{i=1}^{p} 2^{i-1} = 2^p - 1$ rewinds. Since the expected polynomial-time of M is polynomial in k and $p = \mathtt{POLY}(k)$, a hard execution for $V^\star$ can only happen with negligible probability.

Result. Since by Lemma 2, the execution of M is j-deciding for $V^\star$ and by lemma 1, M essentially acts the same when interacting with $P^\star$ in the setup described in $P^\star$'s definition, we can conclude that an execution of M is j-deciding for $P^\star$ as well with overwhelming probability.

Now $P^\star$ has a (non-negligible) probability $\frac{1}{p}$ of correctly guessing $j' = j$ and in such a case, by the definition of a j-deciding execution there will be at least one session where M causes no new view between beginning and end of (a_j, b_j, c_j) when it is solved for the first time. (Specifically there will be at least one session where $\tilde{V}^\star$ outputs $(c_{j'}, view_{\tilde{V}^\star})$ such that the registry $View = view_{\tilde{V}^\star}$.) This means that $P^\star$ will successfully complete at least one session with V. Therefore by soundness of $\langle P, V\rangle$ we have shown that on input a false statement, M will with overwhelming probability output a reject transcript.

To conclude that we have now a deciding algorithm D we need to first deal with the small matter of M running in *expected* polynomial-time rather then strict polynomial-time. This is dealt with in the same way as in [15] (remark 6.1), namely D simply terminates the execution of M after a fixed polynomial number of steps, chosen in such a way that M has a good chance of simulating the conversation.

Thus with proposition 1 and proposition 2 we have shown the efficient algorithm D will, with overwhelming probability, decide correctly contradicting $L \notin \mathcal{BPP}$.

Extension. As we have pointed out, the proof presented in this paper only applies to conversation-based argument systems. We remark that all argument systems known to us are indeed conversation-based. We remark though that our proof can be extended to cover all argument systems by considering an adversarial verifier $V^\star$ that imposes a more sophisticated schedule and rejects with some non-negligible probability the interactions.

4 3-Round RZK Proofs in the UPK Model

In this section we present the first 3-round rZK *proof* (in contrast to argument) system with concurrent soundness for all $\mathcal{NP}$ in the UPK model. Moreover, our construction does not need hardness assumptions with respect to superpolynomial-time algorithms (i.e., complexity leveraging). Similarly to the constant-round zero-knowledge proofs in the plain model of [18], we use at the beginning (i.e., in our case the construction of the public file) an unconditionally hiding commitment scheme. Obviously this can be implemented in one round (therefore in the UPK model) under the assumption that non-interactive unconditionally-hiding commitments exist. This is a strong assumption since so far no construction of non-interactive unconditionally-hiding commitments has been given in the plain model. Therefore, according to the current state-of-the-art, some variations of the model have to be considered. For instance a two-round interactive preprocessing allows to commit with unconditional hiding. This can be alternatively implemented by requiring first a non-interactive set-up stage performed by the provers, then a non-interactive set-up stage performed by the verifiers on input the output of the stage of the provers.

Notice that the hash-based commitment scheme used in the preprocessing stage in [14] actually is a non-interactive unconditionally hiding commitment scheme. However such a scheme can only achieve security with respect to uniform adversarial verifiers [19]. In our construction by using hash-based commitments we obtain the same security achieved in [14].

We stress that the 3-round rZK argument system with concurrent soundness presented in [14] is not a proof system since the verifier sends to the prover a non-interactive zero-knowledge proof of knowledge that is only computational zero knowledge. Moreover, if the proof system of [18] is implemented in 3 rounds in the UPK model, the adversarial verifier can play the second round by choosing one of the polynomially bounded number of public keys and therefore the simulator can not complete the proof with the same probability of the honest prover.

High-Level Overview. The main idea of the previous constructions of resettable zero-knowledge argument systems in the public-key models, is that the simulator obtains (by means of rewinds) the secret key (or at least the output of a computation that needs the secret key) of the verifier and then can perform a straight-line simulation. However, when the adversarial prover is computationally unbounded, on input the public key of the verifier, the prover can compute the secret key and then he could succeed in proving a false statement. We therefore use a different technique. We assume that the verifier generates a public key that corresponds to a super-polynomial number of secret keys each with the same probability. Moreover, we assume that no polynomial-time adversarial verifier can compute more than a bounded polynomial-number of secret keys. A good candidate for this component is a non-interactive unconditionally hiding commitment scheme. Indeed, in this case a commitment corresponds to any possible message of the corresponding message space. This guarantees that even an unbounded prover, by simply looking at the public key of the verifier, does not get any information about the secret key of the verifier. Moreover the binding property guarantees that the polynomial-time adversarial verifier does not find a pair of messages that correspond to the same commitment. This is important for the zero-knowledge property since we will use the fact that the simulator and the verifier will know the *same* secret key. Indeed, the mere possession of one valid secret key does not work since the unbounded prover can compute any secret and therefore could use it for proving false statements. It is therefore necessary to formalize that only knowledge of the specific secret key known by the verifier allows one to simulate the proof without knowledge of the witness for the statement on input.

We formalize this idea by requiring that the prover commits to the witness by means of an unconditionally binding commitment scheme, the verifier sends his secret key and then the prover uses a one-round resettable witness-indistinguishable proof system (e.g., using a ZAP [20], by assuming that the verifier puts in his public key also the first message of the ZAP) for proving that the committed witness, is either a witness for "$x \in L$" or is precisely the secret key sent by the verifier so far (here we use the FLS-paradigm [21]). Intuitively, the protocol is sound with respect to an unbounded prover since when the prover commits to the witness, he has only seen an unconditionally hiding commitment. Therefore he has no advantage for guessing the specific secret key known to the verifier in the set of all possible secret keys. However, notice that the verifier can not use more than once a public key. The protocol is zero-knowledge because the simulator extracts the only secret key that the adversarial verifier knows. Moreover, the fact that the simulator uses a different witness with respect to the one used by the prover in the last proof is not detected by a polynomial-time resetting adversarial verifier since the use of a different witness is plugged in a resettable witness indistinguishable proof system.

The formal protocol considers a bounded polynomial number of public keys since the protocol works in the upperbounded (UPK) public-key model. Since in each session the verifier uses a different secret key, we obtain also concurrent soundness with only 3 rounds.

References

1. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. on Computing **18** (1989) 186–208
2. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. Proc. of STOC ’98, ACM (1998) 409–418
3. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. Proc. of STOC ’00, ACM (2000) 235–244
4. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero-Knowledge Requires $\omega(\log n)$ Rounds. Proc. of STOC ’01, ACM (2001) 570–579
5. Barak, B.: How to Go Beyond the Black-Box Simulation Barrier. Proc. of FOCS ’01, (2001) 106–115
6. Persiano, G., Visconti, I.: Single-Prover Concurrent Zero Knowledge in Almost Constant Rounds. Proc. of ICALP ’05. LNCS, Springer Verlag (2005)
7. Dwork, C., Sahai, A.: Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. Proc. of Crypto ’98. Vol. 1462 of LNCS. (1998) 442–457
8. Goldreich, O.: Concurrent Zero-Knowledge with Timing, Revisited. Proc. of STOC ’02, ACM (2002) 332–340
9. Damgard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. Proc. of Eurocrypt ’00. Vol. 1807 of LNCS (2000) 418–430
10. Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-Interactive Zero-Knowledge. SIAM J. on Computing **20** (1991) 1084–1118
11. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. Proc. of Crypto ’01. Vol. 2139 of LNCS (2001) 542–565
12. Reyzin, L.: Zero-Knowledge with Public Keys. PhD thesis, Massachusetts Institute of Technology (2001)
13. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. Proc. of Crypto ’04. Vol. 3152 of LNCS (2004) 237–253
14. Micali, S., Reyzin, L.: Min-Round Resettable Zero-Knowledge in the Public-key Model. Proc. of Eurocrypt ’01. Vol. 2045 of LNCS (2001) 373–393
15. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. on Computing **25** (1996) 169–192
16. Kilian, J., Petrank, E., Rackoff, C.: Lower Bounds for Zero Knowledge on the Internet. Proc. of FOCS ’98. (1998) 484–492
17. Rosen, A.: A Note on the Round-Complexity of Concurrent Zero-Knowledge. Proc. of Crypto ’00. Vol. 1880 of LNCS (2000) 451–468
18. Goldreich, O., Kahan, A.: How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology **9** (1996) 167–190
19. Reyzin, L.: Personal communication (2005)
20. Dwork, C., Naor, M.: Zaps and their Applications. Proc. of FOCS ’00. (2000) 283–293
21. Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing **29** (1999) 1–28