On Resettably-Sound Resettable Zero Knowledge Arguments ⋆

Yi Deng and Dongdai Lin

The state key laboratory of information security,Institute of software,
Chinese Academy of sciences, Beijing, 100080, China
{ydeng,ddlin}@is.iscas.ac.cn

Abstract. We study the simultaneous resettability problem, namely whether resettably-sound resettable ZK arguments for non-trivial languages exist (posed by Barak et al. [BGGL FOCS'01]), in both the plain model and the bare public-key (BPK for short) model. Under general hardness assumptions, we show:

- in the BPK model, there exist constant-round (*full-fledged*) resettably-sound resettable ZK arguments for NP.
  This resolves a main problem in this model that remained open since the Micali and Reyzin's identification of notions of soundness [MR Crypto 2001] in the BPK model.
- in the plain model, there exist constant-round (*unbounded*) resettably-sound class-bounded resettable ZK (as defined by Deng and Lin in [DL Eurocrypt 2007]) arguments for NP.
  This improves the previous result of Deng and Lin [Eurocrypt 2007] in that the DL construction for class-bounded resettable ZK argument achieves only a *weak* notion of resettable-soundness.

The crux of these results is a construction of constant-round instance-dependent (*full-fledged*) resettably-sound resettable WI *argument of knowledge* (IDWIAOK for short) for any NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$, a notion also introduced by Deng and Lin [Eurocrypt 2007], whose construction, however, obtains only *weak* resettable-soundness when $x_0 \notin L_0$.
Our approach to the simultaneous resettability problem in the BPK model is to make a novel use of IDWIAOK, which gives rise to an elegant structure we call *Σ-puzzles*. Given the fact that all previously known resettable ZK arguments in the BPK model can be achieved in the plain model when ignoring round complexity, we believe this approach will shed light on the simultaneous resettability problem in the plain model.

Keywords. instance-dependent WI, simultaneous resettability, zero knowledge.

1 Introduction

For most cryptographic protocols, it is crucial for an honest party involved in multiple executions of these protocols to refresh its randomness in each run in order to preserve its security. A natural question, of both theoretic and practical interest, is whether it is possible to realize some cryptographic task in such a way that honest party can use a fixed random tape in multiple executions of the resulting protocol without sacrificing its security. Canetti et al. [9] initiated this line of research. In particular, they investigated a central construct in cryptography–zero knowledge proof system [20], and showed that there are proof system for NP that can maintain zero knowledge even when honest prover uses the same randomness tape in polynomially many executions, which gives rise to a strongest notion of zero knowledge, i.e. resettable zero knowledge. Subsequently, in an analogous

⋆ supported by 973 project under Grant No. 2007CB311202, and the National Natural Science Foundation of China under Grant No. 60673069

fashion, Barak et al. [3] put forward the notion of resettable soundness for zero knowledge protocol, which formalizes verfier's security in a scenario where the verifier uses the same randomness tape in polynomially many executions, and showed that Barak's public-coin zero knowledge argument of knowledge [1] can be easily transformed into a resettably-sound zero knowledge argument of knowledge. Barak et al. also posed the following simultaneous resettability problem [3] (Barak et al. also conjectured the answer to this problem is "yes"):

Problem 1: Do there exist resettably-sound resettable ZK arguments for non-trivial languages?

This problem remained open even in the so-called bare public-key (BPK for short) model — a model with very weak setup assumption which simply assumes each verifier registers a public key in a public file before any interaction with the prover begins. This model was also introduced by Canetti et al. [9], aiming at obtaining constant-round resettable zero knowledge argument.

Problem 2: Do there exist resettably-sound resettable ZK arguments for non-trivial languages in the BPK model?

We first stress that both the above two problems are for the case of argument system (rather than proof system) and non-black box zero knowledge. Previous work [3,26] showed that, for non-trivial language, neither resettably-sound zero knowledge proof system nor resettably-sound black-box zero knowledge argument system exists.

Related work. The first attempt to tackle problem 1 was made very recently in [12]. Their main tools are two new instance-dependent primitives: instance-dependent verifiable random functions (we will give more details in section 3) and resettable WI argument of knowledge with instance-dependent *weak* resettable-soundness. For any NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$, The latter one satisfies: 1) resettable witness indistinguishability with respect to any two witnesses w_0 for $x_0 \in L$ and w_1 for $x_1 \in L$; 2) *Weakly* resettably-sound argument of knowledge property when $x_0 \notin L$. That is, the argument of knowledge property when $x_0 \notin L$ holds only against *weak* resetting attack. Deng and Lin defined two types of resetting attack, weak resetting attack and class-bounded resetting attack, both of which are less powerful than fully resetting attack but more powerful than bounded resetting attack [6].

Weak resetting attack: there are a-priori bound on the number of resetting malicious party's distinct first messages.
Class-bounded resetting attack: there are a-priori bounds on both the number of resetting malicious party's distinct first messages and the number of incarnations of honest party with which the malicious resetting party interact.

Theorem 1. [Deng and Lin [12]] Let L_0 and L_1 be two NP languages. If there exist trapdoor permutations and collision-resistant hash functions, then there exists resettable WI argument of knowledge with instance-dependent *weak* resettable-soundness for any NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$.

With the above tools, Deng and Lin presented a transformation of Barak's public-coin bounded concurrent zero knowledge argument into an argument for NP that satisfies both *class-bounded* resettable zero knowledge and *weak* resettable-soundness.

Theorem 2. [Deng and Lin [12]] If there exist trapdoor permutations and collision-resistant hash functions, then there exist class-bounded resettable zero knowledge arguments with *weak* resettable-soundness for all NP languages.

The above results hold in the plain model. Another simple but powerful model that has often been considered for (constant-round) resettable zero knowledge arguments is the BPK model [9], Which merely assumes each verifier deposits a public key in a public file before any interaction with the prover begins. Shortly after its introduction, Micali and Reyzin [22] identified four distinct notions of soundness: one time, sequential, concurrent and resettable-soundness, each of which implies the previous one. Up to now, the strongest notion of soundness that the known constructions for resettable ZK argument in the BPK model achieved is the *concurrent* soundness [16,**?**,**?**].

Previous impossibility result [26] for resettable-soundness in the BPK model was only for the case of *black-box* (resettable or not) zero knowledge argument. Arguments satisfying both resettable ZK and resettable-soundness are not known to exist in the BPK model.

Additional motivation for our study in the BPK model. Besides problem 2 being interesting in itself, we are also motivated by *an interesting fact that all previously known resettable ZK arguments in the BPK model can be achieved in the plain model when ignoring round complexity.* This inspires us to ask whether our feasible result in the BPK model can be further extended to the plain model.

Our contribution. Our first contribution is to present a constant-round instance-dependent (*full-fledged*) resettably-sound resettable WI *argument of knowledge* (IDWIAOK) for any NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$, where L_0 and L_1 are NP languages. That is, we construct an argument for a NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$ that guarantees: 1) resettable witness indistinguishability with respect to any two witnesses w_0 for $x_0 \in L$ and w_1 for $x_1 \in L$, and 2)*fully* resettably-sound argument of knowledge property when $x_0 \notin L$, i.e. when $x_0 \notin L_0$, *argument of knowledge* property holds against any malicious prover mounting *fully* resetting attack. This improves the recent construction for such an argument in which item 2) holds only against *weak* resetting attack.

Lemma 1. Let L_0 and L_1 be two NP languages. If there exist trapdoor permutations and collision-resistant hash functions, then there exists instance-dependent resettably-sound resettable WI argument of knowledge for any NP statement of the form $x_0 \in L_0$ or $x_1 \in L_1$.

This result is obtained by plugging a crucial InstD-VRFs-like component into the Deng and Lin's analogous construction [12]. An immediate consequence is a constant-round class-bounded resettable ZK argument with (*unbounded*) resettable-soundness, which improves the previous result of [12] in that the Deng and Lin's class-bounded resettable ZK argument achieves only *weak* resettable-soundness.

Theorem 3. If there exist trapdoor permutations and collision-resistant hash functions, then there exist resettably-sound class-bounded resettable ZK arguments for NP in the plain model.

We further investigate the simultaneous resettability problem in the BPK model. As aforementioned, what makes this problem interesting is *the fact that all previously known*

resettable ZK arguments in the BPK model can be achieved in the plain model when ignoring round complexity. We construct (constant-round) resettably-sound resettable ZK arguments for NP in the BPK model. This resolves a main problem regarding resettable zero knowledge in this model.

Theorem 4. If there exist trapdoor permutations and collision-resistant hash functions, then there exist constant-round (*full-fledged*) resettably-sound resettable ZK arguments for NP in the BPK model.

Our construction in the BPK model completely departs from the Deng and Lin's paradigm of constructing (restricted version of) simultaneously resettable argument that just uses (restricted version of) IDWIAOK as a key generation protocol (to generate public/secret key pair of verifier' InstD-VRF). Instead, we have both prover and verifier use IDWIAOK to prove that they know answers to some carefully chosen puzzles in turn, which gives rise to an elegant structure we call *Σ-puzzles*. This technique circumvents the current bottleneck (i.e. the bounded concurrency bottleneck of Barak's public-coin zero knowledge argument) and then achieves *full-fledged* simultaneous resettability in the BPK model, and we believe it will also shed light on the simultaneous resettability problem in the plain model.

2 Definitions

Following [12], we give formal definitions of instance-dependent resettably-sound resettable WI argument of knowledge (in the plain model) and resettably-sound resettable ZK argument in the BPK model. Due to space limitations, we omit some related definitions here, such as resettable zero knowledge in the plain model and some bounded versions of this notion, and refer to [3,**?**] for these notions.

Notation. We abbreviate probabilistic polynomial time as PPT. A function $f(n)$ is said to be negligible if for every polynomial $q(n)$ there exists an N such that for all $n \geq N$, $f(n) \leq 1/q(n)$. If L is a language in NP, we define the *associated relation* as the relation $R_L = \{(x, w) \,|\, x \in L; w \text{ is a witness for } `x \in L'\}$.

Resetting and class-bounded resetting attacks. We first recall resetting attack introduced in [9].

Resetting attack [9]. Let $p(\cdot)$ be an arbitrary polynomial, and Let $\overline{x} = x_1, \cdots, x_{p(n)} \in L \cap \{0,1\}^n$ be a sequence of distinct common inputs and $\overline{w} = w_1, \cdots, w_{p(n)}$ be a corresponding witness sequence for $\overline{x}$. The verifier V^*'s resetting attack is defined by the following random process depending on P and V. 1) Randomly pick and fix $p(n)$ random tapes, $r_1, \cdots, r_{p(n)}$, resulting in $p(n)^2$ deterministic incarnations $P^{(i,j)} = P_{x_i, w_i, r_j}$ defined by $P_{x_i,w_i,r_j}(\alpha) = P(x_i, w_i, r_j, \alpha)$, for $(i,j) \in \{1, \cdots, p(n)\} \times \{1, \cdots, p(n)\}$; 2) V^* is allowed to run polynomial many sessions with the $P^{(i,j)}$'s. Throughout those sessions, V^* is allowed to schedule all sessions in interleaving way [1]: V^* can send arbitrary messages to each of the $P^{i,j}$, and obtain the responses of $P^{(i,j)}$ to such messages immediately; 3) Once V^* decides it is done interacting with the $P^{(i,j)}$'s, it produces an output based on its view of the whole interaction. We denote this output by $(P(\overline{w}), V^*)(\overline{x})$.

[1] Actually, in the resettable setting, V^* cannot gain more power from concurrent scheduling than from sequential scheduling (cf. [9]).

Now we recall a restricted version of the resetting attack, i.e., the *class-bounded* resetting attack introduced in [12]. In [12], Deng and Lin categorize all sessions between a verifier and a fixed incarnation of prover into a *class* if they share the *same* verifier's first message msg. A class associated with the incarnation $P^{(i,j)}$ and the verifier's first message msg is donoted $\text{Class}_{P^{(i,j)},\mathsf{msg}}$. *Note that it is possible that a class contains (unbounded) any polynomial number sessions because the verifier is allowed to reset the prover.*

Class-bounded resetting attack.[12] The bounded-class resetting attack is a restricted version of resetting attack in which the malicious verifier V^* is allowed to interact with an *a-priori bounded* number of incarnations and the number of different V^*'s first messages to each incarnation is also a-priori bounded. For instance, a t^3-bounded-class (t be an a-priori fixed polynomial) resetting attack is executed by V^* in the way as defined above, but in which V^* is only allowed to interact with t^2 incarnations of P_{x_i,w_i,r_j}'s, $(i,j) \in \{1,\cdots,t\} \times \{1,\cdots,t\}$, and the number of different V^*'s first messages to each incarnation P_{x_i,w_i,r_j} is a priori bounded by t, where $\overline{x} = x_1,\cdots,x_t \in L \cap \{0,1\}^n$ is a sequence of distinct common inputs and $\overline{w} = w_1,\cdots,w_t$ is a correspondingly witness sequence for $\overline{x}$ as defined in resetting attack, $r_1,\cdots,r_t$ are those provers' random tapes. Note that this results in at most t^3 classes of sessions during the whole execution of this attack.

Definition 1. *(class-bounded resettable ZK argument) Let t be a polynomial, $\overline{x} = x_1,\cdots,x_t \in L \cap \{0,1\}^n$ is a sequence of distinct common inputs and $\overline{w} = w_1,\cdots,w_t$ is a correspondingly witness sequence for $\overline{x}$. An interactive argument (P,V) for a language L is said to be t^3-class-bounded resettable ZK if for every every PPT adversary V^* mounting t^3-class-bounded resetting attack, there exists a PPT M so that $(P(\overline{w}),V^*)(\overline{x})$ and $M(\overline{x})$ are computational indistinguishable.*

Definition 2. *(Resettable WI) Let p be an arbitrary polynomial, L_0 and L_1 be two (possibly the same) NP languages. Let $L = L_0 \vee L_1 = \{(x_0,x_1) : x_0 \in L_0 \text{ or } x_1 \in L_1\}$[2]. An interactive argument (P,V) for language L is said to be resettably witness indistinguishable if for any PPT V^* mounting (*unbounded*) resetting attack, the distribution $(P(\overline{w_0}),V^*)(\overline{x})$ is computationally indistinguishable from $(P(\overline{w_1}),V^*)(\overline{x})$, where $\overline{x} = x^1,\cdots,x^p$, $x^i = (x_0^i,x_1^i) \in L$, $\overline{w_b} = w_b^1,\cdots,w_b^p$ such that $(x_b^i,w_b^i) \in R_{L_b}$ for $1 \le i \le p$, $b \in \{0,1\}$.*

Resettably-sound argument of knowledge. Resetting attack also gives rise to the notion of resettably-sound argument of knowledge introduced in [3].

Definition 3. *(Resettably-sound argument of knowledge.) A resetting attack of a malicious prover P^* on a resettable verifier V is defined by the following random process, indexed by a security parameter n: 1) Uniformly pick and fix $poly(n)$ random-tapes, denoted $r_1,\cdots,r_{poly(n)}$, for V, resulting in deterministic incarnations $V^{(j)}(x) = V_{x,r_j}$, $x \in \{0,1\}^n$ and $j \in \{1,\cdots,poly(n)\}$, defined by $V_{x,r_j}(\alpha) = V(x,r_j,\alpha)$; 2) Taking as input 1^n, P^* is allowed to initiate $poly(n)$ number sessions with the $V^{(j)}(x)$'s, and is allowed to schedule all sessions in interleaving way as usual: P^* can send arbitrary messages to each of the $V^{(j)}(x)$, and obtain the responses of $V^{(j)}(x)$ to such messages immediately.*

We say an argument system (P,V) is a resettably-sound argument of knowledge system if it satisfies:

[2] In our applications, it is sufficient to consider only the "OR" statements.

1. *Resttable-completeness: Considering an arbitrary resetting attack of a PPT P^*. If P^* follows the strategy of P in some sessions after selecting an incarnation $V^{(j)}(x)$ and $x \in L$, then $V^{(j)}(x)$ rejects with negligible probability.*
2. *Resettably-soundness: For every weak resetting attack of a PPT P^*, the probability that in some sessions the corresponding $V^{(j)}(x)$ has accepted a false statement ($x \notin L$) is negligible.*
3. *Argument of knowledge: For every PPT P^*, there exists a PPT machine E such that for every resetting attack of P^*, the probability that E, upon input the description of P^*, outputs a witness for the statement in a session is negligibly close to the probability that P^* convinces V in a session.*

Definition 4. *(Instance-dependent resettably-sound resettable WI) Let L_0 and L_1 be two (possibly the same) NP languages. Let $L = L_0 \vee L_1 = \{(x_0, x_1) : x_0 \in L_0 \text{ or } x_1 \in L_1\}$. An interactive argument (P, V) for language L is said to be instance-dependent resettably-sound resettable WI if it satisfies:*

1. *Resettable WI, as in definition 2.*
2. *For any statement $(x_0, x_1) \in L$, the resettably-sound argument of knowledge property holds when $x_0 \notin L_0$.*

Next, following almost verbatim [22], we give the description of the BPK model and the notions of resettable ZK and resettable-soundness therein.

The BPK Model. The bare public-key model (BPK model) assumes that: 1)A public file F that is a collection of records, each containing a verifier's public key, is available to the prover; 2)An (honest)prover P is an interactive deterministic polynomial-time algorithm that is given as inputs a secret parameter 1^n, a n-bit string $x \in L$, an witness w for $x \in L$, a public file F and a random tape r; and 3)An (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in two stages. In stage one (key generation stage), on input a security parameter 1^n and a random tape r, V generates a key pair (pk, sk) and stores pk in the file F. In stage two (proof stage), on input sk, an n-bit string x and an random string ρ, V performs the interactive protocol with a prover, and outputs "accept x" or "reject x".

Definition 5. *We say that the protocol $< P, V >$ is complete for a language L in $\mathcal{NP}$, if for all n-bit string $x \in L$ and any witness w such that $(x, w) \in R_L$, the probability that V interacting with P on input w, outputs "reject x" is negligible in n.*

Malicious resetting provers and its attacks in the BPK model. Let s be a positive polynomial and P^* be a probabilistic polynomial-time algorithm on input 1^n.

A resetting attack by a s-resetting malicious prover P^* in the BPK model is defined as the following process: 1) Run the key generation stage of V on input 1^n and a random string r to obtain pk and sk (P^* obtains pk and V stores the corresponding sk); 2) Choose $s(n)$ random string ρ_i, $1 \leq i \leq s(n)$, for V; and 3) P^* interacts with oracles for the second stage (proof stage) of the verifier, the i-th oracle having input sk, ρ_i.

Definition 6. *(Resettable soundness in the BPK model) $< P, V >$ satisfies* resettable soundness *for an NP language L in the BPK model if for all positive polynomial s, for*

all s-resetting malicious prover P^*, the probability that in an execution of resetting attack, P^* ever receives "accept x" for $x \notin L$ from any of these oracles is negligible in n

Malicious resetting verifiers and its attacks in the BPK model. Similarly, A resetting attack by a (s,t)-resetting malicious verifier V^*, for any two positive polynomials s and t, can be defined as the following process: 1) In the key generation stage, on input 1^n, V^* receives s instances $x_1, ..., x_{s(n)} \in L$ of length n each, outputs an arbitrary public file F; 2) Choose $r_1, ..., r_{s(n)}$ for P uniformly at random; 2) In proof stage, V^* starts in the final configuration of the key generation stage, is given oracle access to $s^3(n)$ provers, $P(x_i, w_i, pk_j, r_k, F)$, $1 \leq i, j, k \leq s(n)$, and then outputs its view of the interaction: its random string and the messages received form the provers; and 3) The total number of steps of V^* in both stages is at most $t(n)$.

Definition 7. *(Non-black-box resettable ZK in the BPK model) $< P, V >$ is (non-black-box) resettable zero knowledge for an NP language L in the BPK model if for every pair of positive polynomials (s,t), for all (s,t)-resetting malicious verifier V^*, there exists a simulator S, given as input the description of V^*, such that for every $x_1, ..., x_{s(n)} \in L$, the following two distributions are computational distinguishable:*

1. *The output of V^* at the end of a resetting attack described above,*
2. *The output of $S(V^*, x_1, ..., x_{s(n)})$.*

3 How to Construct the Instance-Dependent Resettably-Sound Resttable WI Argument of Knowledge

In this section we show how to construct instance-dependent (*full-fledged*) resettably-sound resttable WI argument of knowledge. This establishes the Lemma 1 and yields Theorem 3.

3.1 The DL approach revisited

A crucial tool in the construction of resettable WI with instance-dependent *weak* resettable-soundness in [12] is the instance-dependent verifiable random functions (InstD-VRF). Informally, an InstD-VRF is a verifiable random function [23] with a special public key, which is generated via a (possibly *interactive*) protocol and contains an instance y with respect to a specific NP language L, but the security requirements on such a function are relaxed: the *pseudorandomness* property[3] is required to hold only when $y \in L$ and the *uniqueness* property is required to hold only when $y \notin L$. y is called the key_instance.

This notion can be realized assuming the existence of trapdoor permutations [12]. The construction is as follows. The querier A and the function owner B execute a key generation protocol and produce an key_instance y, then B selects a pseudorandom function f at random and commits to the description of this function. On input a string a in the domain of f, B returns $f(a)$ and a witness indistinguishable proof (using, say, ZAP [14], in this case the querier should send a setup message for ZAP in the key generation stage) in which he proves that the function value is computed correctly or $y \in L$ using the knowledge of description of f he committed to.

[3] Note that Deng and Lin demonstrate pseudorandomness in a way different from that in [23]: the guessing machine is also provided with correctness proof for the target function value.

The approach to the simultaneous resettability problem suggested in [12] begins with a transformation of resettably-sound bounded concurrent zero knowledge argument $(P_{\mathrm{R}}, V_{\mathrm{R}})$ (which can be obtained from Barak's public-coin bounded concurrent zero knowledge argument, as showed in [3]) into the so-called key_instance-dependent resettably-sound class-bounded resettable zero knowledge argument (DL KInstD ZK argument for short) (P, V) by equipping the verifier in the former system with an InstD-VRF. The Latter argument (P, V) satisfies only the *resettable-soundness* when the key_instance of the InstD-VRF is a YES instance, and satisfies only the *class-bounded resettable zero knowledge* when the key_instance is a NO instance. We now give an informal description of the DL KInstD ZK argument (P, V) proposed in [12].

DL KInstD ZK Argument (P, V) for proving $x \in L$
Phase 1: the key generation protocol KGProt

$V \rightarrow P$ Generate an instance y with respect to a NP language L' (Π_{YES} and Π_{NO} are defined L'), send y and a commitment $c = Com(s, r_s)$ to a random string s (description of a pseudorandom function).

$P \rightarrow V$ Send the first message ρ for a ZAP.

Phase 2: the *Modified* Barak's bounded concurrent ZK argument

$P \Rightarrow V$ P and V execute the Barak's public-coin bounded concurrent zero knowledge protocol with the following exception: in each V's step, instead of sending a random string, V computes $r = f_s(hist)$ ($hist$ is the history of execution of this subprotocol), and sends r plus a proof π in which he proves r is computed correctly or $y \in L'$ by using the prover's strategy in ZAP system. That is, V applies the InstD-VRF specified by $(PK, SK) = ((y, c, \rho), (s, r_s))$ to $hist$, and sends the outputs of InstD-VRF in each V's step. In the V's decision step, V accepts if and only if the transcript of this subprotocol is accepting.

Obviously, the above protocol is not "secure": a malicious resetting verifier will extract a witness for the statement proven when he generates a YES instance in the first step. To obtain some kind of bounded simultaneously resettable argument, we need to find a *secure* (both for the verifier and the prover in the resettable setting) way to generate the instance y for the verifier, instead of letting the verifier itself generate y. The solution suggested in [12] is the so-called resettable WI argument with instance-dependent weak resettable-soundness (hereafter we call it DL WI protocol). With this argument, the DL KInstD ZK argument can be modified into a class-bounded resettable zero knowledge arguments with weak resettable-soundness by having the honest prover generate an NO instance y for the verifier (which serves as the key_instance for the verifier's InstD-VRF that used in the second phase) and use the DL WI protocol to prove that the statement $x \in L$ to be proven is true or y is a YES instance.

The construction of the DL WI protocol employs the classic 3-round WI argument of knowledge [5] as a building block. Let (a, e, z) be the three messages exchanged in an execution of the 3-round argument, and $x_0 \in L_0$ *or* $x_1 \in L_1$ be the statement to be proven. The DL WI protocol proceeds as follows. The verifier first sends a commitment c to a seed of a pseudorandom function and generates the query e by applying this function to the prover's first round message a and uses DL KInstD ZK argument to prove the query e is computed

correctly. In the DL KInstD ZK argument, the instance x_0, serves as the key_instance for a InstD-VRF used by the verifier (the prover in the global system) in the second phase.

DL WI protocol[4] (P, V) **for proving** $x_0 \in L_0$ *or* $x_1 \in L_1$

$V \to P$ Send $c = Com(s)$.

$P \to V$ Send a, the first message of a classic 3-round WI argument.

$V \to P$ Send $e = f_s(x_0, x_1, c, a)$

$V \Rightarrow P$ V proves that e is computed correctly via the DL KInstD ZK protocol in which the x_0 serves as the key_instance for a InstD-VRF used by P (the verifier in the second phase of this subprotocol).

$P \to V$ Send z, the last message of a classic 3-round WI argument.

V's decision V accepts if only if (a, e, z) is accepting.

The DL WI protocol has been shown to be resettable WI. Furthermore, if $x_0 \notin L_0$, it achieves *weak* resettably-sound argument of knowledge property, i.e. when $x_0 \notin L_0$, argument of knowledge property holds only when there is *a-priori bound* on the number of resetting malicious prover's distinct first message.

For readability, we give the whole picture of DL's approach in figure 1.

Bottleneck in achieving *fully* resettable-soundness when $x_0 \notin L_0$–the *bounded concurrent* zero knowledge property of Barak's protocol We first remark that the bottleneck for the DL WI protocol to achieve fully resettable-soundness (argument of knowledge, precisely) when $x_0 \notin L_0$ comes from the bounded concurrent zero knowledge property of the underlying Barak's protocol. Let (P, V) be the DL WI protocol, and the common input are x_0, x_1. Considering the following malicious resetting prover strategy P^*. Upon receiving a specific commitment c, P^* sends polynomial number (not fixed in advance) of distinct first messages a_1,...,$a_{poly(n)}$ (a_i's are the message of the classic 3-round WI argument). After having received all challenges e_1,...,$e_{poly(n)}$, P^*, playing the role of verifier, and V execute the underlying DL KInstD ZK protocol concurrently (or resettably, at P^*'s desire) in which V proves that all these challenges are computed correctly.

Assume the DL WI protocol (P, V) satisfies fully resettably-sound argument of knowledge property when $x_0 \notin L_0$. This means there is an extractor for P^* that can extract an witness for x_1 with essentially the same probability that P^* answers some challenge, say e_i, correctly. It seems that we can construct such an extractor: focusing on the *single* session with prefix of the first three messages c, a_i, e_i, once received z_i, the extractor rewinds to the point just before e_i was sent and resends a random e_i', then uses the simulator guaranteed by DL KInstD ZK protocol to simulate for the false statement (i.e., "e_i is correct"). For all other sessions, the extractor plays as the honest verifier, i.e., always computes e_j, $j \neq i$, by applying the pseudorandom function committed in c to the history. Typically, we justify such an extractor by setting up a hybrid argument, and in one of these hybrid chains, we need to show that if the prover can distinguish it's views in two specific scenarios (that are intermediates between the real interaction and the interaction between the prover and the extractor), we can construct a reduction algorithm, using P^*'s code, to violate the hiding

[4] In fact, the first message of the underlying DL KInstD ZK argument should be sent in the second step of this WI protocol in order to achieve *weak* resettable-soundness. Here we aim at presenting the main idea of the DL WI protocol and therefore ignore this subtle round-message arrangement.

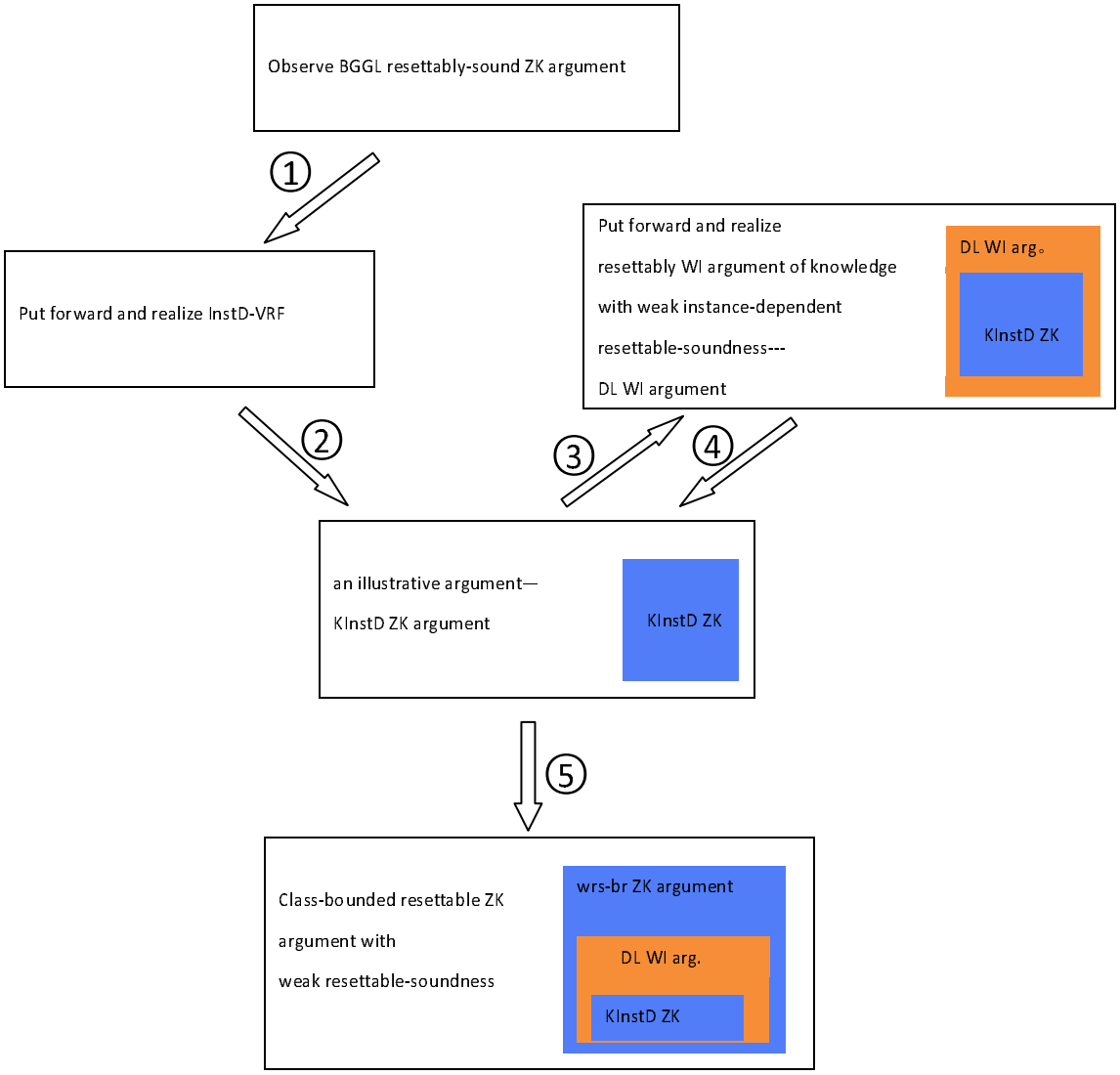

Fig. 1. The approach of [12].

Starting with an observation on BGGL resettably-sound ZK argument, Deng and Lin introduced and realized the notion of InstD-VRF, which naturally leads to an illustrative key_instance-dependent resettably-sound class-bounded resettable ZK argument, i.e. the DL KInstD ZK argument. In order to make this illustrative argument secure against resetting adversaries in the real world, Deng and Lin introduced and realized another new notion of resettable WI argument of knowledge with instance-dependent *weak* resettable-soundness, i.e. the DL WI protocol, which uses DL KInstD ZK argument as a building block. Plugging the DL WI protocol into DL KInstD ZK argument, we get class-bounded resettable ZK arguments with weak resttable-soundness for NP.

property of the commitment scheme which the verifier uses to commit to the pseudorandom

function in the first verifier's step. This requires the reduction algorithm itself can simulate all *unbounded* concurrent (or resettable) executions of DL KInstD ZK protocol without knowing the committed value and the randomness used in c because *all statements "e_i is correct", $1 \leq i \leq poly(n)$ are correlated:* they share the *same* witness, i.e. the committed value and the randomness used in the commitment c. However, the DL KInstD ZK protocol does not meet this requirement: it enjoys only *class-bounded* resettable (or *bounded* concurrent) zero knowledge property due to that its underlying Barak's protocol satisfies only bounded concurrent zero knowledge.

3.2 Constructing Instance-Dependent Resettably-Sound Resettable WI Argument of Knowledge

It seems possible to overcome the above bottleneck by adopting any one of the following approaches:

1. construct public-coin *fully* concurrent zero knowledge argument. Such an argument immediately enables us to construct a DL KInstD ZK protocol satisfying both *fully* resettable zero knowledge on NO key_instance and resettable-soundness on YES key_instance, which meets our requirement.
2. modify the DL WI protocol in such a way that, in all sessions initiated by any resetting prover, those statements "e_i is correct", $1 \leq i \leq poly(n)$ for the (subprotocol) DL KInstD ZK protocol are *uncorrelated*. Note that in this case we do not require the underlying DL KInstD ZK protocol to satisfy *fully* resettable zero knowledge to get a construction of DL WI protocol achieving full resettably-soundness when $x_0 \notin L_0$, instead, as we will see, DL KInstD ZK protocol satisfying 1-class-bounded resettable zero knowledge (when the key_instance is a NO instance) is enough (because to justify soundness we just need to focus on a single class of sessions), and this protocol can be based on the Barak's public-coin *standalone* zero knowledge (i.e. zero knowledge property holds only in the standalone setting) argument.

The first approach raises an apparently hard problem (we do not even know whether such an argument for non-trivial language exists or not). We adopt the second approach to construct an instance-dependent resettably-sound resettable WI argument of knowledge for any NP statement of the form $x_0 \in L_0$ *or* $x_1 \in L_1$. Our construction achieves (unbounded) resettably-sound argument of knowledge property when $x_0 \notin L_0$.

Our protocol is obtained by plugging an extra InstD-VRF-like component into the DL WI protocol. In particular, we have the prover send a commitment c_p to a pseudorandom function as initial message. Upon receiving the verifier's message c_v (it's also a commitment to a pseudorandom function, the first message sent in the DL WI protocol), the prover produces the message a (the first message sent in the underlying 3-round WI protocol) using the randdomness generated by applying the pseudorandom function committed in c_p to the history so far and uses a ZAP to prove that $x_0 \in L_0$ or a is computed correctly. The rest part of our protocol proceeds as the DL WI protocol.

The key idea underlying our construction is that, when $x_0 \notin L_0$, for each verifier's first message c_v, any cheating (resetting) prover cannot produce two different messages a and a' (if V produces c_v by applying some function to c_p). On one hand, this makes all statements (i.e., "the challenge are computed correctly") that the verifier need to prove in the execution

of DL KInstD ZK argument *independent*, and therefore enables us to overcome the obstacle aforementioned and achieve (unbounded) resettably-sound argument of knowledge property. On the other hand, this modification does not endanger the resetable WI property due to two reasons: 1) the prover's message a depends on c_v, and for each c_v, the malicious resetting verifier is bound to one challenge; and 2) when $x_0 \in L_0$, we can produce the message a in arbitrary way and use the witness for $x_0 \in L_0$ to give a valid correct proof. This enables us to present a security reduction from *resettable* WI of our protocol to *concurrent* WI of the underlying 3 round WI argument.

For this idea to work, however, the message a is required to be *uniquely* determined by the randomness used in this step and the common input. We note that not all 3-round WI arguments enjoy this property. Fortunately, the classic parallelized version of Blum's (WI) proof of knowledge for Hamiltonian Cycle satisfies this. The message a of the Blum's protocol is produced in the following way: Given the common input G, the prover chooses a random permutation π and uses a commitment scheme Com to commit to adjacent matrix $M_{\pi(G)}$ of $\pi(G)$ entry by entry. For convenience we use a binary string r_π to represent the permutation π and denote the randomness used in the commitment matrix by r_M (r_M can be also viewed as a matrix in which each entry consists of a random string, and the random string in a entry is used to commit to the value of the corresponding entry in $M_{r_\pi(G)}$), therefore, we have $a = Com(M_{r_\pi(G)}, r_M)$ (committing entry by entry).

The formal description of our construction appears in figure 2. We remark that the order of messages are arranged in a delicate way in order to avoid some (minor) security problem (see also footnote 4). Compared to DL WI protocol, the prover's strategy of our protocol is required to send additional messages c_p, ρ', and τ to guarantee that any *cheating* (resetting) prover cannot produce two different messages a and a' for a verifier's first message c_v in case $x_0 \notin L_0$.

Before proceeding further, we present the extractor and intuition for *unbounded* resettable-soundness property (when $x_0 \notin L_0$) of our construction.

Assume P^* convinces an incarnation $V^j(x)$ on statement $x = (x_0, x_1) \in L$ such that $x_0 \notin L_0$ with probability p in a session. We construct an extractor E as follows.

The extractor E

1. E selects a random string for P^*.
2. E plays the role of honest verifier. Once E obtains an accepting transcript (a, e, z) of the underlying 3 round WI argument in a session in which E emulates the incarnation $V^j_{\mathtt{W}}(x)$, E goes to the next step. Suppose the first three messages exchanged in that session is $(c_p, c_0), (c_v, \rho')$ and (a, τ) (so $e = f_s(x_0, x_1, c_p, c_0, c_v, \rho', a)$, $c_v = Com_v(s, r)$).
3. Assume τ^* was the valid correctness proof for a that appeared for the *first* time in all sessions with the prefix $(c_p, c_0), (c_v, \rho'), a$ (note that τ^* doesn't necessarily equal τ, and this possibly happens due to the resetting attack from P^*). E rewinds P^* to the point where the prefix $(c_p, c_0), (c_v, \rho'), (a, \tau^*)$[5] was first appeared, and then E performs in the following way:

[5] This essentially guarantees E rewinds to the point where $((c_p, c_0), (c_v, \rho'), a)$, rather than $((c_p, c_0), (c_v, \rho'), (a, \tau))$, first appeared. Note that the message a is uniquely determined by the history so far, but τ is not. Also note, for an honest verifier, the message e is uniquely determined by the history *excluding* τ so far, thus, in the extraction, the extractor E must send the same e in all sessions with the

The InstD Resettably-Sound rWI Argument of Knowledge (IDWIAOK)

Common input: two instances $x_0 \in L_0$ *or* $x_1 \in L_1$, a security parameter n, where L_0 and L_1 are NP languages.
The Prover's private input: the witness w such that $(x_0, w) \in R_{L_0}$ or $(x_1, w) \in R_{L_1}$.
Prover's randomness: (r_p^1, r_p^2).
Verifier's randomness: r_v, a seed of a pseudorandom function f_{r_v}

$P \to V$ P sets $(s', r') = f_{r_p^1}(x_0, x_1)$, computes $c_p = Com_p(s', r')$ using a statistically-binding commitment scheme Com_p.
Using the randomness r_p^2, P invokes the DL KInstD ZK argument (it is sufficient to use a KInstD resettable ZK argument that satisfies only 1-class-bounded ZK property) in which P plays the role of verifier, and V will prove to it that the challenge e for the underlying 3-round WI argument is computed correctly, produces the first message c_0 (i.e., the commitment to the description of pseudorandom function) and uses instance x_0 as the key_instance.
Sends c_p,c_0;

$V \to P$ V sets $(r_v^1, r_v^2, \rho') = f_{r_v}(x_0, x_1, c_p, c_0)$. Using the randomness r_v^1, V selects a pseudorandom function $f_s : \{0,1\}^{\leq poly(n)} \to \{0,1\}^{|e|}$ and r, computes $c_v = Com_v(s, r)$ using a statistically-binding commitment scheme Com_v. ρ' will serve as the first message of a ZAP used in next P's step.
Sends c_v, ρ';

$P \to V$ P reduces the instance (x_0, x_1) to a Hamiltonian graph G, and sets $(r_\pi, r_M) = f_{s'}(x_0, x_1, c_p, c_0, c_v, \rho')$, where r_π represents a permutation over the vertices of G, and r_M is a random string matrix as defined before.
P invokes the 3 round WI argument for Hamiltonian Cycle in which it proves G (or, equivalently, $x_0 \in L$ or $x_1 \in L$) has a Hamiltonian cycle, produces the first message $a = Com(M_{r_\pi(G)}, r_M)$ (where Com is also a statistically-binding commitment scheme), and uses the witness s' and r' to prove that $x_0 \in L_0$ or there exist s' and r' such that $c_p = Com_p(s', r') \wedge (r_\pi, r_M) = f_{s'}(x_0, x_1, c_p, c_0, c, \rho') \wedge a = Com(M_{r_\pi(G)}, r_M)$ via a ZAP (with the first message ρ'). Let the second message (the proof) be τ.
Sends a, τ;

$V \to P$ V computes $e = f_s(x_0, x_1, c_p, c_0, c_v, \rho', a)$ (notice that we exclude the proof τ from the history to which the f_s is applied to generate e, and this is crucial to justify the soundness). Using the randomness r_v^2, V selects the first message ρ for a ZAP used in DL KInstD ZK argument.
Sends e, ρ;

$V \Rightarrow P$ P and V continue to run DL' KInstD ZK argument in which V proves there exist s, r such that $e = f_s(x_0, x_1, c_p, c_0, c, \rho', a)$ and $c_v = Com_v(s, r)$. The public key for P's InstD-VRF (P plays the role of the verifier in this execution of DL KInstD ZK argument) is $PK = (x_0, c_0, \rho)$ and the corresponding secret key is the decommitment to c_0.

$P \to V$ Sends the answer z to the query e according to the 3 round WI argument for Hamiltonian Cycle if the above transcript is accepting.

V**'s Decision** V accepts if only if the transcript (a, e, z) is accepting.

Fig. 2. The instance-dependent resettably-sound resettable WI argument of knowledge.

– For those sessions having the same prefix $((c_p, c_0), (c_v, \rho'), a)$ with *arbitrary* valid correctness proof for a) , E chooses another query $e' \neq e$ randomly, and sends e' to P^*, then runs the simulator guaranteed by DL KInstD ZK argument (that satisfies only 1-class-bounded resettable zero knowledge property when the key_instance is a NO instance) and give proofs that e' is computed correctly.
Note that all executions of DL KInstD ZK argument in those sessions having the

same first three messages $(c_p, c_0), (c_v, \rho')$ and a (excluding τ), and this requires E rewinds to point where $(c_p, c_0), (c_v, \rho'), (a, \tau^*)$ (τ^* as defined above), rather than $((c_p, c_0), (c_v, \rho'), (a, \tau))$, first appeared.

same prefix $((c_p, c_0), (c_v, \rho'), a)$ fall into one class according to the terminology in [12], that is, all these executions are carried out with the *same* incarnation of verifier $V^j(x)$ under the *same* P^*'s (the verifier in DL KInstD ZK argument) first message c_0 of DL KInstD ZK argument.

- For all other sessions, E acts as honest verifier.

4. E repeats the step 3 until it obtains an accepting transcript (a, e', z') with $e \neq e'$.
5. E computes the witness for x_1 (note that we assume $x_0 \notin L$) from the two transcripts (a, e, z) and (a, e', z'), outputs it.

Intuition for *unbounded* resettable-soundness when $x_0 \notin L$. First of all, keep in mind that the DL KInstD ZK argument used in our protocol depicted in fig.1 satisfies only 1-class-bounded resettable zero knowledge property (when the key_instance is a NO instance), which can be obtained from the known Barak's public-coin **standalone** zero knowledge argument.

Note that in E's step 3, the statements "the challenge is correct" in the two categories of sessions (i.e. sessions with prefix $((c_p, c_0), (c_v, \rho'), a)$ and sessions with prefix $((c'_p, c'_0), (c'_v, \rho''), a')$, $((c_p, c_0), (c_v, \rho'), a) \neq ((c'_p, c'_0), (c'_v, \rho''), a'))$, are (almost) uncorrelated: since $((c_p, c_0), (c_v, \rho'), a) \neq ((c'_p, c'_0), (c'_v, \rho''), a'))$, it must be the case $(c_p, c_0) \neq (c'_p, c'_0)$[6], and this leads to the pseudorandom functions committed in c_v by $V^j(x)$ in these two categories of sessions are (almost) independent and thus the statements "the challenge is correct" in the two categories of sessions are (almost) uncorrelated.

This crucial observation enables us to employ DL KInstD ZK argument satisfying only 1-class-bounded resettable zero knowledge property (for verifier to prove "the challenge is correct") is that all executions of DL KInstD ZK argument in those sessions the same prefix $((c_p, c_0), (c_v, \rho'), a)$ fall into a *single* class (namely, the class specified by the *same* incarnation of verifier $V^j(x)$, the prover in DL KInstD ZK argument, and the *same* P^*'s (the verifier in DL KInstD ZK argument) first message c_0 of DL KInstD ZK argument), and E needs only to simulate this class of sessions and plays as honest verifier in any other sessions (that are uncorrelated to sessions in the class which E needs to simulate) in straight line way.

We now prove that the argument depicted in Fig.1 is an instant-dependent resettably-sound resettable WI argument of knowledge (i.e., satisfying the definition 4). This follows from the following two claims and establishes Lemma 1.3.

Claim 3.1 The argument depicted in Fig.1 satisfies resettable witness indeistinguishability as defined in definition 2.

proof. Let $L = L_0 \vee L_1 = \{(x_0, x_1) : x_0 \in L_0 \text{ or } x_1 \in L_1\}$, $poly(\cdot)$ be an arbitrary polynomial and V^* be an arbitrary malicious PPT verifier strategy mounting resetting attack. Let $\overline{x} = x^1, \cdots, x^{poly(n)}$, $x^i = (x_0^i, x_1^i) \in L$, $\overline{w_b} = w_b^1, \cdots, w_b^{poly(n)}$ such that $(x_b^i, w_b^i) \in R_{L_b}$ for $i = 1, \cdots, poly(n)$, $b = 0, 1$. We set up hybrid experiments, in which the distribution

[6] Note that for the deterministic $V^j(x)$, if $(c_p, c_0) = (c'_p, c'_0)$, then we have $(c_v, \rho') = (c'_v, \rho'')$. By the fact that message a is uniquely determined by the first two round messages (for simplicity we consider the same common input $x = (x_0, x_1) \in L$) when $x_0 \notin L_0$, so if $((c_p, c_0), (c_v, \rho')) = ((c'_p, c'_0), (c'_v, \rho''))$, then $a = a'$. Thus we have $((c_p, c_0), (c_v, \rho'), a) = ((c'_p, c'_0), (c'_v, \rho''), a')$, which contradicts the condition $((c_p, c_0), (c_v, \rho'), a) \neq ((c'_p, c'_0), (c'_v, \rho''), a')$

in one of the hybrids is indistinguishable from that in preceding one, to prove the *Resettable witness indistinguishability*.

Hybrid 0 The distribution $(P(\overline{w_0}), V^*)(\overline{x})$ (V^*'s view in interaction with honest prover using $\overline{w_0}$ as witnesses).

Hybrid 1 The distribution $(P_{1,\overline{w_0}}(\overline{w_0}), V^*)(\overline{x})$, where $P_{1,\overline{w_0}}$ follows the P's strategy except that for every i, $1 \leq i \leq poly(n)$, $P_{1,\overline{w_0}}$ uses the witness w_0^i for x_0^i to execute the ZAP in P's second step in which it proves $x_0^i \in L_0$ or the a is computed correctly (in Hybrid 1, Hybrid 2 and Hybrid 3, we use the subscript $\overline{w_0}$ to indicate that the prover will use the witness in the sequence $\overline{w_0}$ to execute the ZAP in its second step). Note that in Hybrid 0 the honest prover always uses the witness for the statement that a is computed correctly to prove that $x_0^i \in L_0$ or the a is computed correctly (see the protocol in figure 1). So, we can claim that this hybrid is indistinguishable from the Hybrid 0 due to the witness indistinguishability of the ZAP.

Hybrid 2 The distribution $(P_{2,\overline{w_0}}(\overline{w_0}), V^*)(\overline{x})$, where $P_{2,\overline{w_0}}$ follows $P_{1,\overline{w_0}}$'s strategy, except that it selects a pseudorandom function $f_{s''}$ at random (independent of $f_{s'}$ that committed in its first message c_p) and produces a in the P's second step using randomness generated by applying this function to the history so far, and for all session shared the same c_p, P_2 always use the same function $f_{s''}$. This hybrid is indistinguishable from the Hybrid 1 due to computationally hiding of the commitment scheme Com_p.

Hybrid 3 The distribution $(P_{2,\overline{w_0}}(\overline{w_1}), V^*)(\overline{x})$, where $P_{2,\overline{w_0}}$, given both $\overline{w_0}$ (used to execute the ZAP in its second step) and $\overline{w_1}$, follows prover's strategy in Hybrid 2, except that for all i, $1 \leq i \leq poly(n)$, it uses w_1^i for x_1^i to execute the underlying 3-round WI argument for Hamiltonian Cycle . We will show that if there is a distinguisher can tell this hybrid and Hybrid 2, then there exists a PPT verifier strategy in concurrent model that violates the witness indistinguishability of the underlying 3-round WI argument. Note that witness indistinguishability is preserved in concurrent model, thus we conclude that this hybrid is indistinguishable from the Hybrid 2. Detailed proof follows shortly.

Hybrid 4 The distribution $(P_{3,\overline{w_0}}(\overline{w_1}), V^*)(\overline{x})$, where $P_{3,\overline{w_0}}$ follows $P_{2,\overline{w_0}}$'s strategy, except that it produces a in the same way that the honest prover does, that is, in the P's second step it uses randomness generated by applying $f_{s'}$ that committed in its first message c_p to the history so far to produce a (still, $P_{3,\overline{w_0}}$ uses the witness in the sequence $\overline{w_0}$ to execute the ZAP in its second step). Again, the indistinguishability between this hybrid and Hybrid 3 is due to computationally hiding of the commitment scheme Com_p.

Hybrid 5 $(P(\overline{w_1}), V^*)(\overline{x})$. Note that the prover P in this hybrid follows the honest prover's strategy with the witness sequence $\overline{w_1}$, and therefore the only difference between the P's strategy and the prover's strategy $P_{3,\overline{w_0}}$ in Hybrid 4 is that, for all i, $1 \leq i \leq poly(n)$, P uses the witness s' and r' that is used in its first step to compute c_p ($c_p = Com_p(s', r')$) to prove $x_0 \in L_0$ or a is computed correctly via a ZAP in its second step, while the prover's strategy $P_{3,\overline{w_0}}$ in Hybrid 4 uses the witness w_0^i for x_0^i. The indistinguishability between this hybrid and Hybrid 4 is due to the same reason for the indistinguishability between Hybrid 0 and Hybrid 1, i.e., the witness indistinguishability of the ZAP.

Let $(P_{\mathtt{W}}, V_{\mathtt{W}})$ be the underlying 3-round WI argument for Hamiltonian Cycle. Now we show that Hybrid 2 is indistinguishable from Hybrid 3. Assume otherwise, there exists an al-

gorithm D distinguishes the two distributions $((P_{2,\overline{w_0}}(\overline{w_0}), V^*)(\overline{x})$ and $((P_{2,\overline{w_0}}(\overline{w_1}), V^*)(\overline{x})$, then we can construct a PPT $V_{\mathtt{W}}^*$ in concurrent model, such that two distributions $(P_{\mathtt{W}}(\overline{w_0}), V_{\mathtt{W}}^*)(\overline{x})$ and $(P_{\mathtt{W}}(\overline{w_1}), V_{\mathtt{W}}^*)(\overline{x})$ are distinguishable. This contradicts the witness indistinguishability of $(P_{\mathtt{W}}, V_{\mathtt{W}})$ (note that WI holds even in concurrent model, cf [17]).

$V_{\mathtt{W}}^*$, given $(\overline{w_0})$ as input[7], incorporates V^* and handles V^*'s messages as follows. 1) When V^* initiates a session with incarnation $P_{2,\overline{w_0}}^{i,j}$ $(1 \leq i, j \leq poly(n))$, $V_{\mathtt{W}}^*$ computes c_p and c_0 as the honest prover does, and replies with c_p and c_0 internally. 2) When V^* sends a kth new first message (i.e., c_v and ρ') to incarnation $P_{2,\overline{w_0}}^{i,j}$ $(1 \leq i, j \leq poly(n))$, $V_{\mathtt{W}}^*$ initiates a session with $P_{\mathtt{W}}^{i,j_k}$ (defined by $P_{\mathtt{W}}^{i,j_k}(\alpha) = P_{\mathtt{W}}(x^i, w^i, r_{j_k}, \alpha)$, where $x^i = (x_0^i, x_1^i)$, $w^i = (w_0^i, w_1^i)$, r_{j_k}'s are selected independently), obtains the $P_{\mathtt{W}}^{i,j_k}$'s first message a, uses the witness w_0^i to produce the proof τ that the first message a of the DL KInstD ZK argument is computed correctly, stores a and τ and then forwards them to V^*; 3) when V^* sends a query e to $P_{2,\overline{w_0}}^{i,j}$ for its kth first message a, $V_{\mathtt{W}}^*$ stores it and continues the execution of the DL KInstD ZK argument, and once V^* accepts this proof, forwards the query e to $P_{\mathtt{W}}^{i,j_k}$, stores its response z and forwards it to V^*. All V^*'s repeated messages are replied with the same answer.

Noth that if V^* does not send different challenges to $P_{2,\overline{w_0}}^{i,j}$ regarding the same $P_{2,\overline{w_0}}^{i,j}$'s (actually produced by $P_{\mathtt{W}}^{i,j_k}$) first message a, then $V_{\mathtt{W}}^*$ works in concurrent model (i.e., it holds at most one session with each incarnation $P_{\mathtt{W}}^{i,j_k}$). Furthermore, if all incarnations of $P_{\mathtt{W}}$ use the witness sequence $\overline{w_b} = w_b^1, \cdots, w_b^{poly(n)}$ in above interaction, the V^*'s view in the above experiment is identical to $((P_{2,\overline{w_0}}(\overline{w_b}), V^*)(\overline{x})$ (note the fact that both $P_{2,\overline{w_0}}$ and $V_{\mathtt{W}}^*$ use the same witness in the sequence $\overline{w_0}$ to produce the proof τ), and furthermore, notice that the $V_{\mathtt{W}}^*$'s view (i.e., $(P_{\mathtt{W}}(\overline{w_b}), V_{\mathtt{W}}^*)(\overline{x})$) is just the copy of V^*'s view, Thus, we conclude if D can distinguish $((P_{2,\overline{w_0}}(\overline{w_0}), V^*)(\overline{x})$ and $((P_{2,\overline{w_0}}(\overline{w_1}), V^*)(\overline{x})$, it also can distinguish $(P_{\mathtt{W}}(\overline{w_0}), V_{\mathtt{W}}^*)(\overline{x})$ and $(P_{\mathtt{W}}(\overline{w_1}), V_{\mathtt{W}}^*)(\overline{x})$.

We claim the probability that $V_{\mathtt{W}}^*$ sends different challenges to $P_{\mathtt{W}}^{i,j_k}$'s first message a is negligible because the underlying DL KInstD ZK argument satisfies resettably-soundness when $x_0 \in L$, this completes the proof that Hybrid 2 is indistinguishable from Hybrid 3. ■

Claim 3.2 When $x_0 \notin L_0$, the argument depicted in Fig. 2 satisfies resettably-sound argument of knowledge property as defined in definition 3.

proof. We give this proof by justifying the extractor showed in subsection 3.2. The success of the extractor (when $x_0 \notin L_0$) relies on this key observation: the verifier's first message (c_v, ρ') will determine an *unique* first message a of the underlying 3-round WI argument when $x_0 \notin L_0$, this enables the analysis of indistinguishability between the Hybrid 1 and the Hybrid 2 (see below).

Suppose that P^* convinces an incarnation $V^j(x)$ on statement $x = (x_0, x_1)$ such that $x_0 \notin L$ with probability p in a session.

We first note that if the probability P^* convinces $V^j(x)$ is non-negligible, then E will run in expected polynomial time.

[7] Note that WI is required to hold against malicious verifiers that take both $(\overline{w_0})$ and $(\overline{w_1})$ as the auxiliary input

In step 3, all simulated proofs fall into *one class* with respect to underlying the DL KInstD ZK argument[8], i.e., the class specified by the first verifier (the prover in global system) message c_0 (contained in the first prover's message (c_p, c_0)) and the incarnation of prover $V_{\mathtt{W}}^j(x)$ (the verifier in global system), so the simulation will be run successful due to the 1-class-bounded resettable ZK property of the underlying DL KInstD ZK argument.

There are two differences between P^*'s view simulated by E in step 3 and that generated in a real interaction: 1) $e' \neq f_s(x_0, x_1, c_p, c_0, c_v, \rho', a)$ in E's step 3, 2) in sessions in which the first three messages (excluding τ) exchanged equals $(c_p, c_0), (c_v, \rho'), a$, all proofs given by E are simulated.

If we prove that P^*'s view in above two scenarios are indistinguishable, then the extractor E will extract a witness for $x_1 \in L$ with probability negligibly close to p. This completes the proof.

We claim the P^*'s view in above two scenarios are indistinguishable despite these differences. This follows from the hybrid experiments below. It is easy to see that P^*'s view in one of the hybrids is indistinguishable from that in preceding one (Here we assume P^* only interacts with the incarnation $V^j(x)$ for simplicity. Note that different incarnations of the verifier will choose pseudorandom functions independently).

Hybrid 0 P^* interacts with the incarnation $V^j(x)$.

Hybrid 1 P^* interacts with V_1, where V_1 follows the $V^j(x)$'s strategy (computes challenges of the underlying 3-round WI argument in honest way in all sessions) but in all those sessions having the same first three messages (excluding τ) $(c_p, c_0), (c_v, \rho')$ and a, V_1 runs the simulator for the DL KInstD ZK argument to give proofs that the challenge in those sessions (note that all those sessions will share the same challenge due to the way by which the challenge is produced) are computes correctly . The P^*'s view in this hybrid is indistinguishable from that in Hybrid 0 due to the 1-class-bounded ZK property of the DL KInstD ZK argument when $x_0 \notin L_0$. Note that all simulated proofs fall into *one class* with respect to underlying DL KInstD ZK argument.

Hybrid 2 P^* interacts with V_2, where V_2 follows V_1's strategy, but chooses a pseudorandom function $f_{s'}$ randomly (independent of f_s, the one that V_2 committed to), computes challenge e by using $f_{s'}$ in all those sessions having the same first three messages (excluding τ) $(c_p, c_0), (c_v, \rho')$ and a, and runs the simulator for the DL KInstD ZK argument to give proofs that those queries generated by using $f_{s'}$ are computes correctly. The P^*'s view in this hybrid is indistinguishable form that in Hybrid 1 due to the hiding property of the statistically-binding commitment scheme Com_v which $V^j(x)$ used to committed to the description of f_s. Otherwise, by setting up a standard experiment, we can use the distinguisher to break the hiding property of Com_v.

We emphasize that, in such a experiment, we need to simulate all proofs with respect to the target commitment c_v (for which we guess the value (s or s') that is committed to in c_v), fortunately, this can be done due to the following two reasons: 1) all these proofs fall in one class (this is crucially different from the case in [12] in which there are

[8] This happened also to the extractor constructed in [12], however, it is worthy of notice that this fact cannot help the construction of the DL WI protocol to achieve unbounded resettably-sound argument of knowledge when $x_0 \notin L$ because we have to simulate many other sessions (classes) related to the same verifier's commitment in the analysis of the extractor.

possibly many different classes with respect to the same target commitment, and though the extractor presented in [12] needs only to simulate one class among those different classes as our extractor, the analysis of their extractor requires all those different classes can be simulated). Note that the a verifier's first message (c_v, ρ') will determine an *unique* first message a of the underlying 3-round WI argument when $x_0 \notin L_0$, and this is crucial for our argument to achieve higher level of security than the one in [12]. 2) the underlying DL KInstD ZK argument satisfies 1-class-bounded ZK property when $x_0 \notin L_0$.

Hybrid 3 P^* interacts with E in step 3. Note that the only difference between the strategy of E in step 3 and V_2 lies in the fact that in all those sessions having the same first three messages $(c_p, c_0), (c_v, \rho')$ and a (excluding τ), E sends a random challenge e' while V_2 computes challenge e by using $f_{s'}$. P^*'s view in this hybrid is indistinguishable form that in Hybrid 2 due to the pseudorandomness of $f_{s'}$. ■

4 Proof of Theorem 1.4

The construction of IDWIAOK presented in last section yields the Theorem 1.4 immediately. By simply replacing DL WI protocol with our IDWIAOK in the construction of DL's class-bounded resettable ZK argument with *weak* resettable-soundness presented in section 5 in [12], we get a (*unbounded*) resettably-sound class-bounded resettable ZK argument. In particular, our protocol is derived from DL KInstD ZK protocol (see subsection 3.1) with the following modification: in the phase 1 (the key generation phase), instead having the verifier himself generate the key_instance for his InstD-VRF, we have the honest prover generate a NO key_instance y for the verifier's InstD-VRF (used in the second phase) and use our IDWIAOK to prove that the statement $x \in L$ to be proven is true or y is a YES instance.

The reason that the resulting protocol achieves *unbounded* resettable-soundness is simply due to the *unbounded* resettable-soundness of the IDWIAOK. The security analysis of this protocol is essentially identical to the one of DL's class-bounded resettable ZK argument with *weak* resettable-soundness [12], and so is omitted here.

5 Σ-Puzzles: settling the simultaneous resettability problem in the BPK Model

So far, the IDWIAOK (and the DL WI protocol) has been only used to setup a key_instance for the verifier's InstD-VRF in the DL paradigm for simultaneously resettable argument. As we have seen, this paradigm essentially suffers from the *bounded-concurrency* bottleneck of the Barak's public-coin argument in achieving *fully* resettable zero knowledge[9].

This bottleneck exists even in the BPK model. Considering the following adaption of our (*unbounded*) resettably-sound class-bounded resettable ZK argument (P, V) into the BPK model: we have V register its first message as its public key. Note that this adaption does not affect the resettable-soundness. However, the resulting protocol still *fails* to achieve

[9] Note that our construction of IDWIAOK just shows how to avoid this bottleneck in achieving full resettable-soundness.

fully resettable ZK, though it does achieve incarnation-bounded resettable ZK (i.e., there are a-priori bound on the number of incarnations of honest prover with which the malicious resetting verifier interact), a property stronger than class-bounded resettable ZK.

We now make a novel use of IDWIAOK, which circumvents the current technical bottleneck and gives rise to a elegant construction (we call it Σ-Puzzles) for *full-fledged* simultaneously resettable argument in the BPK model.

5.1 Σ-Puzzles Protocol in the BPK Model: Intuition and Subtlety

Let $f : \{0,1\}^n \to \{0,1\}^n$ be an one-way function, $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator. Consider the following 3 phases "Σ-Puzzles" protocol for statement $x \in L$ ($|x| = n$).

A Σ-Puzzles protocol in the BPK model (informal)
Common input: $x \in L$
The public key: a puzzle β ($\beta = f(\alpha)$)
P's private input: a witness w for $x \in L$
V's private input: α, the solution to β.

$P \Longrightarrow V$ P chooses a random string γ of length $2n$ (served as a puzzle specified by G) and sends it to V.
Using IDWIAOK, P proves that $x \in L$ or he knows a solution δ to the puzzle γ, i.e., $\gamma = G(\delta)$.

$V \Longrightarrow P$ Using IDWIAOK, V proves that he knows a solution δ to the puzzle γ or a solution α to the puzzle β.

$P \Longrightarrow V$ Using IDWIAOK, P proves that $x \in L$ or he knows the solution α to the puzzle β.

As usual, We assume that the resettably-sound argument of knowledge property of every IDWIAOK used in the above protocol depends on the first instance corresponding to the common input.

Intuition. The above protocol seems to be both resettable ZK and resettably-sound.

To establish *resettable ZK*, the simulator just needs to extract the solution to puzzle β (that serves as the public key): in every session, it generates a YES instance γ (i.e., $\exists\delta$ s.t. $\gamma = G(\delta)$) and uses δ as witness to execute the first IDWIAOK, and using the solution to puzzle β extracted from malicious verifier as witness, it will complete the last IDWIAOK successfully. This simulation strategy seems to work: 1) the first and the last IDWIAOK are resettable WI; 2) the extraction can be done by using the extractor associated with IDWIAOK (*even though γ is a YES instance*[10]) in the second phase; and 3) the puzzles (i.e., public keys) for which the simulator needs to extract the corresponding solutions are fixed in advance.

To establish *resettable-soundness*, we can construct an algorithm B that break the one-wayness of f in the following way: B first extracts the solution to the puzzle γ from the

[10] It seems that the extractor described in last section works only in case the first instance γ is NO instance. However, as we will see, assuming language L_0 (defined by G) is a hard-to-decide, we can also do extraction even when $\delta \in L_0$ in our case in which the YES instance δ is generated by the simulator (that plays the role of verifier in second IDWIAOK). This is implied by the indistinguishability between Hybrid 2, Hybrid 3, and Hybrid 4 in the proof of **Claim 5.1** presented in next section.

malicious resetting prover P^* in the first IDWIAOK, and uses this solution to complete the second IDWIAOK, then it extracts the solution to the puzzle β in the third IDWIAOK (i.e., finds the preimage of β). B seems to work due to the fact that the first and the last IDWIAOK satisfy resettably-sound argument of knowledge property when $x \notin L$ (note that x serves as the first instance of the common inputs for the first and the last IDWIAOK).

Subtlety. Indeed, the above intuition for resettable zero knowledge works. Using a mix of black-box simulation strategy and non-black-box extraction strategy, combined with an idea in [21], we can extract the answers to puzzles registered by a malicious resetting verifier *one by one*, and then do simulation successfully.

However, the above intuition for resettable-soundness is problematic. Observe that in the extraction process described in section 3, while focusing on a *single* "class", the extractor needs to simulate the honest verifier in other sessions. Thus, when a IDWIAOK is embedded in a big argument system, a successful extraction with respect to this IDWIAOK requires:

It is easy to simulate the honest party, which acts as the verifier in this IDWIAOK, in an executions of the big system.

This is not the case with our extraction used to justify resettable-soundness. For the breaking algorithm B, which plays the role of the verifier in the first IDWIAOK and does extraction in this phase, it is impossible to complete the second IDWIAOK without knowledge of the solution to the puzzle γ[11] (note that B does not know the preimage of β).

To use the malicious resetting prover P^*'s power to invert f, the breaking algorithm B need to simulate the verifier in all sessions initiated by P^*. This requires B to extract solution to every puzzle γ generated by P^* in every session. Note that this task cannot be done in polynomial time *even* when P^* is a concurrent adversary due to two facts: 1) the extraction requires rewinding; and 2) P^* can generate these γ's based on transcripts of other sessions. Similar in spirit to concurrent scheduling that causes "nest effect" in [15], we can construct a concurrent adversary P^* such that B cannot do this simulation in polynomial time.

we need some new idea to overcome the above obstacle in analysis of resettable-soundness.

A Remark on Non-Malleability. Another security concern that arises in our setting is malleability[11]. The malicious party (prover or verifier) may be able to mount a man-in-the-middle attack, for instance, the malicious verifier initiates two sessions scheduled in a delicate way, and may cheat in the second IDWIAOK (in which he plays the prover) in session two based on some information obtained from the first IDWIAOK (in which he plays the verifier) in session one. We note that the basic 3-round WI argument—if we implement it using some number-theoretic assumptions—is malleable as described in [2] (the authors of [2]called it "divertible"): the man-in-the-middle can produce a first message a in a session in which he plays the prover (the right session) based on another first message a' he received from another session in which he plays the verifier (the left session), and upon receiving the challenge e in the right session, he computes a challenge e' based on e in a specific way. At last, once the man-in-the-middle received a accepting last message z' in the left

[11] A similar case happens to the simulator, however, as we will see in detailed proof, the simulator works due to the fact that all puzzles registered by malicious verifier are *fixed in advance*.

session, he can produce an accepting last message z in the right session without knowing the witness for the common input. However, this attack does not apply to our case (intuitively) due to the fact that once a' is sent by the honest party, the challenge e' has already been uniquely determined by the history of this session up to a' (note that both Com_v and Com_p are statistically-binding commitments), and the man-in-the-middle can not change it at his desire. We will give a rigorous treatment of this problem in our security proof. (implied by **Claim 5.1** presented in next section).

We argue that in the construction of IDWIAOK does not require any underlying commitment scheme to be non-malleable, and we do not claim that our IDWIAOK is non-malleable. What we claim is that our argument in the BPK model (presented in next section)is secure (i.e., satisfying simultaneous resettability) against any possible man-in-the-middle attack mounted by a *resetting verifier* (or *resetting prover*).

5.2 Resettably-Sound Resettable ZK Argument in the BPK Model

We use *signature* to deal with the aforementioned subtlety. In the key generation stage, we have the verifier publish a public key of a signature scheme that satisfies existential unforgeability against adaptive chosen message attack; In proof stage, we have the verifier prove that he knows a solution δ to the puzzle γ or a valid signature on the message (x, γ) (x is the statement to be proven) in the second IDWIAOK, and have the prover prove that $x \in L$ or he knows a valid signature on the message (x, γ) in the third IDWIAOK. To prove the resulting protocol enjoys resettable-soundness, we construct an algorithm that breaks existential unforgeability of the signature scheme with probability negligibly close to the one that the malicious resetting prover can cheat.

The crux of this idea is that the breaking algorithm needs only to focus on those sessions with the same prover's first message γ on input the false statement $x \notin L$ in which it intends to extract a valid signature on message (x, γ), and, *along with the help from signing oracle* (thus it can obtain all valid signatures on messages that does not equal (x, γ)), plays the role of the verifier in straight line way for all other sessions. On the other hand, as we will see, this modification does not compromise the resettable ZK property (though it increases the running time of the simulator up to a polynomial factor) due to two facts: 1) both the public key sequence $(ver\_k_1, ..., ver\_k_s)$ and the statement sequence $x_1, ..., x_{s(n)} \in L$ are fixed in advance, and 2) all γ_k's are determined *only* by the randomness tape of the prover, and therefore can also be viewed as fixed in advance when all provers are assumed to be honest in the proof of zero knowledge.

Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator, and $SS = (KG, Sig, Ver)$ be a signature scheme against adaptive chosen message attack, in which KG, Sig, and Ver denote key-generation algorithm, signing algorithm, and verifying algorithm respectively. We depict the protocol in Fig. 2.

Hardness assumptions. Note that IDWIAOK is based on the existence of trapdoor permutations and hash functions, and both the signature scheme against adaptive chosen message attack and the pseudorandom generator can be constructed from any one-way functions. So, the existence of trapdoor permutations and hash functions are sufficient for our resettably-sound resettable ZK argument in the BPK model.

The Resettably-Sound Resttable ZK Argument (P, V) in the BPK model

Common input: $x \in L$ ($|x| = n$), the public file F, an index i that specifies the i-th entry $pk_i = ver\_k_i$.
P**'s private input**: the witness w such that $(x, w) \in R_L$.
V**'s private input**: sig_k_i such that $(sig\_k_i, ver\_k_i) = KG(1^n)$, where KG is a key generation algorithm of the signature scheme $SS = (KG, Sig, Ver)$.
P**'s randomness**: (γ, r_p), where γ is chosen uniformly at random and $|\gamma| = 2n$.
V**'s randomness**: r_v.

$P \longrightarrow V$ P sends γ;

$P \Longrightarrow V$ P and V execute the IDWIAOK in which P proves that $x \in L$ or there exists δ such that $\gamma = G(\delta)$ by using w as witness;

In this stage, P sets $r_p^1 = f_{r_p}(x, ver\_k_i, \gamma)$ for the random tape of the prover strategy in IDWIAOK, and V sets $r_v^1 = f_{r_v}(x, ver\_k_i, \gamma)$ for the random tape of the verifier's strategy in IDWIAOK.

$V \Longrightarrow P$ V and P execute the IDWIAOK in which V proves that there exists δ such that $\gamma = G(\delta)$ or there exists σ such that $Ver(ver\_k_i, \sigma, (x, \gamma)) = 1$ (i.e., σ is a valid signature on (x, γ)). In this execution of IDWIAOK, V uses the secret key sig_k_i to produce a valid signature σ and uses it as witness;

In this stage, P sets $r_p^2 = f_{r_p}(x, ver\_k_i, \gamma, tran_1)$ for the random tape of the verifier's strategy in IDWIAOK, and V sets $r_v^2 = f_{r_v}(x, ver\_k_i, \gamma, tran_1)$ for the random tape of the prover's strategy in IDWIAOK, where $tran_1$ is the transcript of execution of the first IDWIAOK.

$P \Longrightarrow V$ P and V execute the IDWIAOK in which P proves that $x \in L$ or there exists σ such that $Ver(ver\_k_i, \sigma, (x, \gamma)) = 1$ by using w as witness.

In this stage, P sets $r_p^3 = f_{r_p}(x, ver\_k_i, \gamma, tran_1, tran_2)$ for the random tape of the prover's strategy in IDWIAOK, and V sets $r_v^3 = f_{r_v}(x, ver\_k_i, \gamma, tran_1, tran_2)$ for the random tape of the prover's strategy in IDWIAOK, where $tran_2$ is the transcript of execution of the second IDWIAOK.

Fig. 3. The resettably-sound resettable ZK argument for NP language L.

5.3 Analyzing our protocol

In this section, We show the protocol depicted in fig. 2. satisfies both resettably-soundness and resettable ZK. This establishes Theorem 1.5.

Completeness is straightforward.

Resettable ZK. Let V^* be an (s, t)-resetting malicious verifier. Assume, in real world, On input a fixed YES instance sequence $x_1, ..., x_{s(n)} \in L$ of length n each, V^* generates its public keys $(ver\_k_1, ..., ver\_k_s)$, and interacts with $s^3(n)$ incarnations of prover, $P(x_i, w_i, ver\_k_j, r_k, F)$, $1 \leq i, j, k \leq s(n)$, where r_k's are chosen independently and uniformly at random. Without loss of generality, we also assume that V^* sends a first message (i, j, k) to initiate a session with incarnation $P(x_i, w_i, ver\_k_j, r_k, F)$. We construct a simulator S as required by definition 2.6.

S operates as follows. First, given a fixed YES instance sequence $x_1, ..., x_s \in L$ of length n each as input, S runs the key-generation phase of V^* to obtain the public file F consists of s entries $(ver\_k_1, ..., ver\_k_s)$.

In proof stage, Upon receiving the initiation message (i, j, k), S chooses δ_k randomly and sends $\gamma_k = G(\delta_k)$ to V^*, then uses δ_k as witness to complete the first IDWIAOK. Note that if S manages to extract a valid signature $\sigma_{(i,j,k)}$ (under the public key ver_k_j) on the message (x_i, γ_k) (i.e., $Ver(ver\_k_j, \sigma_{(i,j,k)}, (x_i, \gamma_k)) = 1$) from V^* in the second IDWIAOK,

then S can succeed in simulating the prover $P(x_i, w_i, ver\_k_j, r_k, F)$ by using this valid signature as witness in the execution of the third IDWIAOK. Note that when all common inputs $(x_1, ..., x_s)$ are YES instance, the first and the third IDWIAOK enjoy resettable WI property.

S proceeds in sequential phases in proof stage. In each phase, S either obtains a new valid signature or completes the whole simulations unless it detects some V^*'s cheating behavior (then it aborts as the honest prover). *We argue that S will complete the whole simulation in at most $s^3 + 1$ phases.* Notice that both the public key sequence $(ver\_k_1, ..., ver\_k_s)$ and the statement sequence $x_1, ..., x_s \in L$ are fixed in advance, and all γ_k's are determined *only* by the randomness tape of the prover. Therefore, for s^3 incarnations $P(x_i, w_i, ver\_k_j, r_k, F)$, $1 \leq i, j, k \leq s$, knowledge of s^3 valid signatures $\sigma_{(i,j,k)}$ enables a successful simulation.

The remaining task is to show how the simulator extract a valid signature in one phase. Such an extraction seems a bit *unusual*: the simulator S needs to extract a witness in the second IDWIAOK in which the first instance γ is a YES instance (note that S must generate a YES instance γ to complete the first IDWIAOK), and the second IDWIAOK does NOT guarantee such an extractor at all when the first instance γ is a YES instance! However, we will show it is feasible to do such a extraction due to the fact γ is a hard-to-decide instance and is generated by the prover (hence the verifier does not know the witness to γ, though it is a YES instance).

Now we present the strategy of S in one phase. We begin with some notations and terminology. Recall that a session is an interaction with an incarnation $P(x_i, w_i, ver\_k_j, r_k, F)$. Due to the resetting attack, V^* may initiate many (possibly incomplete) interactions with the same incarnation $P(x_i, w_i, ver\_k_j, r_k, F)$. We denote all interactions with $P(x_i, w_i, ver\_k_j, r_k, F)$ by $\mathsf{Int}_{(i,j,k)}$. We say $\mathsf{Int}_{(i,j,k)}$ *solved* if S has obtained a valid signature $\sigma_{(i,j,k)}$ (under the public key ver_k_j) on the message x_i, γ_k, otherwise we call it *unsolved*. A session is called *solved* if it belongs to a *solved* $\mathsf{Int}_{(i,j,k)}$ for some (i, j, k).

S in one phase

1. S selects a random tape for V^*, and plays the role of prover. Like the honest prover, S checks whether V^*'s message is accepting in each step, if not, S aborts this session immediately.
2. Throughout this phase, S adopts the following trivial strategy in every *solved* session: generate a YES instance γ and complete the first IDWIAOK using γ's witness, then play the role of honest verifier in the second IDWIAOK and use the valid signature that S has obtained as witness to execute the third IDWIAOK.
3. For every *unsolved* session, S generates a YES instance γ, completes the first IDWIAOK using γ's witness, and plays the role of honest verifier in the second IDWIAOK until one session reaches the last step of the second IDWIAOK and the transcript of this session so far is accepting.

   Assume that this session belongs to interaction $\mathsf{Int}_{(i,j,k)}$ (i.e., the statement proven, the first prover's message, and the public key are x_i, γ_k, and ver_k_j respectively.). Suppose $tran$ and $(c_p, c_0), (c_v, \rho'), (a, \tau)$ are the transcript up to the end of the run of the first IDWIAOK and the first three messages exchanged in the second IDWIAOK in this session respectively, (a, e, z) is the accepting transcript of the underlying 3 round

WI argument in the second IDWIAOK, and τ^* (not necessarily equal τ) is the valid correctness proof (for a) that appeared for the *first* time in all sessions with the prefix $tran, (c_p, c_0), (c_v, \rho'), a$.

4. Once S received the above accepting transcript (a, e, z), it rewinds to the point where the prefix $(tran, (c_p, c_0), (c_v, \rho'), (a, \tau^*))$[12] (no matter what the correctness proof for a is. We call this point **rewinding point**) was first appeared, and then, for the first two IDWIAOKs in every *unsolved* session, S performs in the following way:
   - For all those *unsolved* sessions in $\mathsf{Int}_{(i,j,k)}$ with the prefix $(tran, (c_p, c_0), (c_v, \rho'), a)$ (we call such sessions **target** sessions), S chooses another query $e' \neq e$ randomly, and sends e' to V^* as the challenge of the underlying 3 round WI argument in the second IDWIAOK, then runs the simulator for DL KInstD ZK argument and give proofs that e' is computed correctly.
     Notice that all executions of DL KInstD ZK argument in **target** sessions fall into one class according to the terminology in [12].
   - For every other *unsolved* session, S, as usual, generates a YES instance γ, completes the first IDWIAOK using γ's witness, and plays the role of honest verifier in the second IDWIAOK.
5. If one of those **target** sessions is the *first* one that reaches the last step of the second IDWIAOK and the new transcript (a, e', z') of the underlying 3 round WI argument in the second IDWIAOK is accepting, S halts and computes the witness $\sigma_{(i,j,k)}$ from the two accepting transcripts (a, e, z) and (a, e', z'), and stores it. Otherwise, S goes to step 4.

We first remark that there are two differences between the extraction strategy used by S in step 4 and the one used by the extractor for IDWIAOK: 1) the underlying (1-class-bounded) DL KInstD ZK argument does NOT guarantee a simulator in case γ_k is a YES instance (which is the first instance of the common input for the second IDWIAOK and serves as the key_instance for the V^*'s InstD-VRF in this DL KInstD ZK argument). This is the very reason why IDWIAOK does not provide resettably-sound argument of knowledge property when the first instance of the common input is a YES instance. 2) the strategy used by S requires that in the second run (after rewinding), one of the **target** sessions reaches the last step of the second IDWIAOK again *before* any other *unsolved* session reaches this step. Note that if an unsolved **non-target** session reaches the this step first in the second run of S, S will possibly get stuck: it will not be able to complete the third IDWIAOK successfully because it knows neither the relevant valid signature nor the witness for the statement to be proven in the global system.

Nevertheless, as mentioned, due to the fact that γ_k is a hard-to-decide instance and is generated by the prover, we are able to show DL KInstD ZK argument is still simulatable in

[12] As done in the extraction presented in section 3.2, see also footnote 3. Here we stress that the three messages $((c_p, c_0), (c_v, \rho'), a)$ of the second IDWIAOK will never appears in any session with different prefix $tran'$ due to the way of generating random bits in each step. Note that this is crucial for justifying our simulator S: Once the prefix $tran', (c_p, c_0), (c_v, \rho'), a$ appeared in a **non-target** (i.e., $tran' \neq tran$) session, S is supposed to send the challenge e to V^* immediately, then in a **target** session, any different challenge $e' \neq e$ resent by S will be detected by V^* because S is bound to send only *one* correct challenge with the same history $((c_p, c_0), (c_v, \rho'), a)$ of the second IDWIAOK.

our case. Moreover, observe that once DL KInstD ZK argument is simulatable in our case, the requirement in item 2) can be met without causing the running time of S to increase much.

Now we give a formal analysis of the simulator. Consider S in a specific phase. Let $A = \{(\mathsf{Int}_{(i',j',k')})|\mathsf{Int}_{(i',j',k')}$ *has been solved before* S *enters this phase.*$\}$. Let Real^A be real interactions till the point where one session (i.e. one of the **target sessions** as defined in the description of S) outside A reaches the last step of the second IDWIAOK. Assume that this session is in $\mathsf{Int}_{(i,j,k)}$, i.e., the statement proven, the first prover's message, and the public key are x_i, γ_k, and ver_k_j respectively. Let Sim^A_{fst} be the first run of S in this phase, i.e., simulation before rewinding, and Sim^A_{sec} be the second run of S in this phase, which consists of two parts: simulation till the **rewinding point** and its continuation carried out in step 4 (after rewinding).

The resettable zero knowledge property of our protocol follows from the following three claims.

Claim 5.1 Both Sim^A_{fst} and Sim^A_{sec} are indistinguishable from Real^A.

This claim guarantees S will extract a witness to the statement "there exists δ_k such that $\gamma_k = G(\delta_k)$ or there exists $\sigma_{(i,j,k)}$ such that $Ver(ver\_k_j, \sigma_{(i,j,k)}, (x_i, \gamma_k)) = 1$" (i.e., either δ_k or $\sigma_{(i,j,k)}$) with essentially the same probability that V^* convinces the honest prover that this statement is true in the second IDWIAOK in one of the **target** sessions.

Claim 5.2 S extracts $\sigma_{(i,j,k)}$ (rather than δ_k) with probability negligibly close to the probability that V^* gives a successful proof in the second IDWIAOK in one of the **target** sessions.

Claim 5.3 S halts in expected polynomial time.

proof of **Claim 5.1.** We mainly prove that Sim^A_{sec} is indistinguishable from Real^A. The fact that Sim^A_{fst} is indistinguishable from Real^A can be easily shown in the same way and we omit it here.

Again, we perform a hybrid argument, in which each hybrid is indistinguishable from its preceding neighbor, to establish this claim. In what follows, interactions defined in all hybrids are carried out till the point where one session outside A reaches the last step of the second IDWIAOK, and the **target** sessions are defined as in the description of S.

Hybrid 0 Real^A.

Hybrid 1 Interactions HSim^A_1 between H_1 and V^*, in which H_1, given all witnesses $(w_1, ..., w_{s(n)})$ as input, follows the honest prover strategy except that, in **target** sessions[13], it runs the simulator for DL KInstD ZK argument to give a proof that the challenge is correct in the second IDWIAOK.

Due to the 1-class-bounded (as mentioned, all executions of DL KInstD ZK argument in **target** sessions fall into one class) resettable ZK property of DL KInstD ZK argument

[13] Actually, H_1 cannot tell whether a session is a **target** session before a session outside A reaches the last step of the second IDWIAOK, but this does not matter: H_1 can find **target** sessions by following the honest verifier strategy in all sessions outside A. Once one of these sessions reached the last step of the second IDWIAOK (at this point **target** sessions are defined), then H_1 rewinds and uses the strategy defined in this hybrid in those **target** session. In fact, HSim^A_1 is defined as the second run of H_1, and this holds for the following H_2, and H_3. For simplifying presentation, we ignore this problem totally.

when γ is a NO instance (note that all γ's in this interactions are NO instances), HSim_1^A is indistinguishable from Real^A.

Hybrid 2 Interactions HSim_2^A between H_2 and V^*, in which H_2, given all witnesses $(w_1, ..., w_{s(n)})$ as input, follows H_1's strategy except that he sends a random challenge e' in the second $\mathsf{IDWIAOK}$ (and runs the simulator to prove that e' is correct, as H_1) in those **target** sessions.

Due to the hiding property of Com_v, HSim_2^A is indistinguishable from HSim_1^A.

Hybrid 3 Interactions HSim_3^A between H_3 and V^*, in which H_3, given all witnesses $(w_1, ..., w_{s(n)})$ as input, follows the H_2 strategy except that he generates a YES instance γ as its first message in every session.

Due to the pseudorandomness of G, HSim_3^A is indistinguishable from HSim_2^A.

Hybrid 4 Sim_{sec}^A. Note that the only difference between HSim_3^A and Sim_{sec}^A is that, in Sim_{sec}^A, S uses the witness corresponding to the second part of the statement in execution of the first $\mathsf{IDWIAOK}$ and third $\mathsf{IDWIAOK}$ in every session.

Due to resettable WI property of these first and third $\mathsf{IDWIAOK}$s, Sim_{sec}^A is indistinguishable from HSim_3^A. ■

Why Claim 5.1 implies security against man-in-the-middle attack by V^*. The reason that our protocol is secure against man-in-the-middle attack mounted by V^* is that, once V^* has received a first message of the underlying 3-round WI of the first or third $\mathsf{IDWIAOK}$ in a session, V^* cannot produce challenge e for this session adaptively based on some other session's transcript, *even* in the case that S shows to V^* how to simulate on a false statement "e' is correct" in one of **target** sessions. This in turn is due to:

- The DL KInstD ZK argument in the second $\mathsf{IDWIAOK}$ (which is used by S to prove "challenge e' is correct") in any **target** session satisfies 1-class-bounded resttable ZK even though the key_instance γ_k for the $\mathsf{InstD\text{-}VRF}$ hold by V^* is a YES instance. This is implies by the indistinguishability between HSim_3^A and HSim_2^A.
- The DL KInstD ZK argument in the first or third $\mathsf{IDWIAOK}$ (which is used by V^* to prove "challenge e is correct") in any session satisfies resettable-soundness because all key_instances (i.e., the common inputs x_i, $1 \leq i \leq s$) for those $\mathsf{InstD\text{-}VRFs}$ hold by S are YES instance. This implies the indistinguishability between Sim_{sec}^A and HSim_3^A.

proof of **Claim 5.2.** Consider the following two modified simulators:

Simulator S_1 , given all witnesses $(w_1, ..., w_{s(n)})$ as input, follows S's strategy except that it always uses the witness for the common input of the current session (one of $(w_1, ..., w_{s(n)})$) to execute the first and third $\mathsf{IDWIAOK}$.

Simulator S_2 , given all witnesses $(w_1, ..., w_{s(n)})$ as input, follows S_1's strategy except that it generates a random string γ as its first message in every session.

It is easy to see that, except for negligible probability, S and S_1 extract the same witness for the statement "there exists δ_k such that $\gamma_k = G(\delta_k)$ or there exists $\sigma_{(i,j,k)}$ such that $Ver(ver\_k_j, \sigma_{(i,j,k)}, (x_i, \gamma_k)) = 1$"[14], due to the resettable WI property of these first

[14] Here all witnesses to the same instance of this statement is regarded as the same

and third IDWIAOKs. The same holds for S_1 and S_2: they will extract the same witness for the above statement except for negligible probability, due to the pseudorandomness of G.

Note that all γs generated by S_2 are NO instances except for exponentially small probability, hence S_2 extracts $\sigma_{(i,j,k)}$ with the same probability that V^* convinces the prover that the above statement is true in the second IDWIAOK in one of the **target** sessions. Thus the claim follows. ■

proof of **Claim 5.3.** We observe that if with probability p one of **target** sessions is the first that reaches the last step of the second IDWIAOK and the transcript of this session (so far) is accepting, then, after rewinding, the probability that one of the **target** sessions is still the *first* session that reaches the last step of the second IDWIAOK is negligibly close to p (this observation was also used in [21] and in [9]), otherwise, we can use V^* to break either the computation hiding of the commitment scheme Com_v or the (1-class-bounded) resettable ZK property of DL KInstD ZK argument (this property holds even when γ_k is a YES instance, see Appendix B for detailed proof).

Assume that the expected time of the interactions between S and V^* before rewinding is $poly(n)$, which is a polynomial. So if S obtained an accepting transcript of a **target** session before rewinding, S will find a new valid signature during a phase in time $poly(n)/p$. Thus, the expected running time of S in one phase is $(1-p)\cdot poly(n)+p\cdot(poly(n)/p)$, thus S will complete the whole simulation in expected time $(s^3+1)[(1-p)\cdot poly(n)+p\cdot(poly(n)/p)]$, which is a polynomial. ■

Resettable-soundness. This property is proved by contradiction. Assume that there is a PPT P^* that can cheat an honest verifier with non-negligible probability. We construct an algorithm B that, having access to a signing oracle, breaks the existential unforgeability of the signature scheme SS with non-negligible probability.

Assume in real world, on input a false statement x and a public key (ver_k), P^* interacts with s incarnations of verifier, $V(x, sig\_k, \rho_i)$, $1 \le i \le s$, where ρ_i's are chosen independently and uniformly at random. We denote by Int_k the set of all sessions having the first P^*'s first message γ_k. Note that for two sessions having different P^*'s first message γ, even initiated by P^* with the *same* incarnation $V(x, sig\_k, \rho_i)$, they look like sessions between P^* and different incarnations of verifier due to the fact that $V(x, sig\_k, \rho_i)$ applies a pseudorandom function to history to generate randomness in each session. Assume P^* sends at most t different γ's, $\gamma_1, ..., \gamma_t$. where t is a polynomial. B guesses an Int_j in which P^* will cheat an honest verifier. For any session in Int_k, $k \neq j$, B, under the help from signing oracle, acts as the honest verifier; For session in Int_j, B rewinds P^* in the first IDWIAOK, obtains δ_j such that $\gamma_j = G(\delta_j)$ and uses it to complete the second IDWIAOK, at last, B rewinds P^* in the last IDWIAOK and will obtains a valid signature σ on new message (x, γ_j).

The breaking algorithm B

1. B selects a random string for P^*, and plays the role of verifier. Like the honest verifier, B checks whether P^*'s message is acceptable in each step, if not, B aborts this session immediately.
2. B uniformly chooses j from $\{1, ..., t\}$.

3. Throughout this breaking process, B adopts honest verifier strategy in every session in Int_k, $k \neq j$. In the execution of the second IDWIAOK, B obtains a valid signature on message (x, γ_k) from the signing oracle, and uses it to complete the second IDWIAOK.
4. Once a session in Int_j reaches the last step of the second IDWIAOK and the transcript of this session so far is accepting, B goes to next step.

   Suppose that $(c_p, c_0), (c_v, \rho')$, and (a, τ) are the first three messages exchanged in the first IDWIAOK in this session, (a, e, z) is the accepting transcript of the underlying 3 round WI argument in the first IDWIAOK, and τ^* (not necessarily equal τ) is the valid correctness proof (for a) that appeared for the *first* time in all sessions with the prefix $tran, (c_p, c_0), (c_v, \rho'), a$.
5. B rewinds to the point where the prefix of a session $(\gamma_j, (c_p, c_0), (c_v, \rho'), (a, \tau^*))$ (As done in the extraction presented in section 3.2, see also footnote 3.) was first sent, and then, during the execution of the first IDWIAOK, B performs in the following way:
   - For every session in Int_j having the prefix $(\gamma_j, (c_p, c_0), (c_v, \rho'), a)$ (no matter what the correctness proof for a is. As before, we call such sessions **target** sessions), B chooses another query $e' \neq e$ randomly, and sends e' to P^* as the challenge of the underlying 3 round WI argument in the first IDWIAOK, then runs the simulator for DL KInstD ZK argument and give proofs that e' is computed correctly[15].
   - For every other session in Int_j, B acts as honest verifier in the first IDWIAOK.
6. If one of those **target** sessions is the *first* one that reaches the last step of the first IDWIAOK and the new transcript (a, e', z') of the underlying 3 round WI argument in the second IDWIAOK is accepting, B computes the witness δ_j (such that $\gamma_j = G(\delta_j)$) from the two accepting transcripts (a, e, z) and (a, e', z'), stores it, and goes to next step. Otherwise, B goes to step 5.
7. For all sessions in Int_j, B uses δ_j as witness to execute the second IDWIAOK, and acts as honest verifier in the third IDWIAOK.
8. Upon receiving a complete accepting transcript of a session in Int_j, B uses the same strategy as the extractor E presented in section 3 to extract a valid signature σ on new message (x, γ_j). Suppose $tran$, and $((c_p, c_0), (c_v, \rho'), (a, \tau))$ are the transcript before the third IDWIAOK and the first three messages exchanges in the third IDWIAOK respectively, and (a, e, z) be the accepting transcript of the underlying 3 round WI argument in the third IDWIAOK. In particular, B rewinds to the point where $(tran, (c_p, c_0), (c_v, \rho'), a)$ was first sent, sends another query $e' \neq e$ and proves e' is correct by using simulator guaranteed by DL KInstD ZK argument, and acts as honest verifier in other sessions in Int_j except that it uses δ_j as witness to execute the second IDWIAOK. Once (a, e', z') is obtained, B computes the witness σ on the message (x, γ_j) that used in the execution of third IDWIAOK by P^* (note that we assume $x \notin L$) from the two transcripts (a, e, z) and (a, e', z'), outputs it and halts.

Note that when $x \notin L$, the simulation for DL KInstD ZK argument in step 5 and 8 will be successful, moreover, by standard hybrid argument, we can conclude that whole P^*'s view

[15] Again, note that all executions of DL KInstD ZK argument in these target sessions fall into one class, and the three messages $((c_p, c_0), (c_v, \rho'), a)$ of the first IDWIAOK will never appears in any session with different prefix γ_i for $i \neq j$ due to the way of generating random bits in each step.

during the above breaking process is indistinguishable from the real interactions. Thus, if P^* can cheat with non-negligible probability p, then B will halt in expected polynomial time and output an signature on the message (x, γ_j) for which it does not query the signing oracle with probability p/t (B guesses the correct number j with probability $1/t$), which is still a non-negligible probability. This contradicts the existential unforgeability of the signature scheme SS.

6 Discussion

It is interesting to note that the resettable ZK property of all previous arguments in BPK model is demonstrated via *black-box* simulation (in [3,13], non-black-box technique are used only to demonstrate soundness), and, when ignoring the round complexity, all previously known feasible results in BPK model can be achieved in the plain model by increasing the "rewinding opportunity" for simulator (we can view the second IDWIAOK as a rewinding opportunity for simulator). By modifying our protocol in the BPK model in the same way, we get the following protocol.

Protocol for statement $x \in L$.

preamble

$V \longrightarrow P$ V sends many, say $m = logn$, public keys $ver\_k_1, ..., ver\_k_m$;

$P \Longrightarrow V$ P sends a random string γ.
Using IDWIAOK, P proves to V that $x \in L$ or he knows δ such that $\gamma = G(\delta)$;

from $i = 1$ to m

$V \Longrightarrow P$ Using IDWIAOK, V proves to P that he knows δ such that $\gamma = G(\delta)$ or a valid signature σ_i such that $Ver(ver\_k_i, \sigma_i, (x, \gamma)) = 1$;

main body

$P \Longrightarrow V$ Using IDWIAOK, P proves to V that $x \in L$ or he knows σ_i such that $Ver(ver\_k_i, \sigma_i, (x, \gamma)) = 1$ for some $i \in \{1, ..., m\}$.

We do not know whether the above protocol satisfy simultaneous resettability. if yes, it seems we need a simulation strategy more sophisticated than the (black-box) simulation strategy as described in [25].

Acknowledgements We are grateful to two anonymous referees from Crypto'08 for their encouragement and positive comments, and thanks to all referees for their suggestions to improve the presentation of this paper.

Yi deng thanks Ivan Visconti and Pino Persiano for many valuable discussions.

References

[1] B. Barak. How to go beyond the black-box simulation barrier. In Proc. of IEEE FOCS 2001, pp.106-115.

[2] M. Burmester, Y. Desmedt. All Languages in NP Have Divertible Zero-Knowledge Proofs and Arguments Under Cryptographic Assumptions. In Advances in Cryptology-Eurocrypt'90, LNCS 473, pp.1-10, 1991.

[3] B. Barak, O. Goldreich, S. Goldwasser, Y. Lindell. Resettably sound Zero Knowledge and its Applications. In Proc. of IEEE FOCS 2001, pp. 116-125.

[4] B. Barak, O. Goldreich. Universal Arguments and Their Applications. In Proc. of IEEE CCC 2002, pp. 194-203.
[5] M. Blum. How to Prove a Theorem so No One Else can Claim It. In Proc. of ICM'86, pp. 1444-1451, 1986.
[6] B. Barak, Y. Lindell, S. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. In Proc. of IEEE FOCS 2003, pp.384-393
[7] M. Blum, S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo Random Bits. In Proc. of IEEE FOCS 1982, pp. 112-117
[8] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In Proc. of IEEE FOCS 2001, pp.136-145
[9] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali. Resettable Zero Knowledge. In Proc. of ACM STOC 2000, pp.235-244
[10] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Concurrent Zero-Knowledge requires $\Omega(logn)$ rounds. In Proc. of ACM STOC 2001, pp.570-579.
[11] D. Dolev, C. Dwork and M. Naor. Non-malleable Cryptography. SIAM J. on Computing 30(2):391-437, 2000.
[12] Yi Deng, Dongdai Lin. Instance-Dependent Verifiable Random Functions and Their Application to Simultaneous Resettability. In Advances in Cryptology-Eurocrypt'07, LNCS4515, pp.148-168, 2007.
[13] Yi Deng, Dongdai Lin. Resettable Zero Knowledge in the Bare Public-Key Model under Standard Assumption. Cryptology ePrint Archive, Report 2006/239.
[14] C. Dwork, M. Naor. Zaps and Their Applications. In Proc. of IEEE FOCS 2000, pp.283-293
[15] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In Proc. of ACM STOC 1998, pp.409-418.
[16] G. Di Crescenzo, Giuseppe Persiano, Ivan Visconti. Constant Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In Advances of Cryptology-Crypto'04, Springer LNCS3152, pp.237-253
[17] U.Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In Proc. of ACM STOC 1990, pp.416-426.
[18] O. Goldreich. Foundation of Cryptography-Basic Tools. Cambridge University Press, 2001.
[19] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proof systems. J. ACM, 38(3), pp.691-729, 1991.
[20] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM. J. Computing, 18(1):186-208, February 1989.
[21] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. In Proc. of ACM STOC'00, pp.683-692, 2003.
[22] S. Micali, L. Reyzin. Soundness in the Public-Key Model. In Advances in Cryptology-CRYPTO'01, Springer LNCS2139, pp.542-565.
[23] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In Proc. of IEEE FOCS'99, pp. 120-130, 1999.
[24] M. Naor. Bit Commitment using Pseudorandomness. Journal of Cryptology 4(2): 151-158, 1991.
[25] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent Zero-Knowledge with Logarithmic Round Complexity. Manoj Prabhakaran, Alon Rosen and Amit Sahai. Concurrent Zero-Knowledge with Logarithmic Round Complexity. In Proc. of IEEE FOCS'02, pp.366-375, 2002.
[26] L. Reyzin. Zero Knowledge with Public Keys. Ph.D thesis, 2001.
[27] A. Yao. Theory and Applications of Trapdoor Functions. In Proc. of IEEE FOCS'82, pp.80-91, 1982.
[28] M. Yung, Y. Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In Advances in Cryptology-Eurocrypt'07, LNCS4515, pp.129-147, 2007.