Instance-Dependent Verifiable Random Functions and Their Application to Simultaneous Resettability*

Yi Deng[1] and Dongdai Lin[1]

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 introduce a notion of *instance-dependent verifiable random functions* (InstD-VRFs for short). Informally, an InstD-VRF is, in some sense, a verifiable random function [23] with a special public key, which is generated via a (possibly)*interactive* protocol and contains an instance $y \in L \cap \{0,1\}^*$ for a specific NP language L, but the security requirements on such a function are relaxed: we only require the *pseudorandomness* property when $y \in L$ and only require the *uniqueness* property when $y \notin L$, instead of requiring both pseudorandomness and uniqueness to hold *simultaneously*. We show that this notion can be realized under standard assumption.

Our motivation is the conjecture posed by Barak et al.[2], which states there exist resettably-sound resettable zero knowledge arguments for NP. The instance-dependent verifiable random functions is a powerful tool to tackle this problem. We first use them to obtain two interesting instance-dependent argument systems from the Barak's public-coin bounded concurrent zero knowledge argument [1], and then, we

1. Construct the *first* (constant round) zero knowledge arguments for NP enjoying a *certain* simultaneous resettability under standard hardness assumptions in the plain model, which we call bounded-class resettable ZK arguments with weak resettable-soundness. Though the malicious party (prover or verifier) in such system is limited to a kind of bounded resetting attack, We put NO restrictions on the number of the total resets made by malicious party.
2. show that, under standard assumptions, if there exist public-coin concurrent zero knowledge arguments for NP, there exist the resettably-sound resetable zero knowledge arguments for NP.

Keywords. instance-dependent verifiable random functions, simultaneous resettability, zero knowledge.

1 Introduction

Pseudorandom functions, introduced by Goldreich, Goldwasser and Micali [14], are basic cryptographic primitives and have been used in a wide range of cryptographic applications. Loosely speaking, pseudorandom functions are efficient

* This work is supported by the National Natural Science Foundation of China under Grant No. 60673069

functions that cannot be tell apart from truly random functions by any polynomial-time observer that given a black-box access to those functions.

In some applications, the seed (the description of a specific pseudorandom function) owner needs to convince the observer (querier) that his reply to the the observer's query is correctly computed in order to protect the observer. To serve this need, Micali et al. put forward the concept of verifiable random functions [23]. Informally, a verifiable random function is described by a public/secret key pair, and its output consists of two part, a pseudorandom value and a proof that proving this value is correct. The security requirements for such a function are: 1)uniqueness. Except with a exponentially small probability, there is only one value for a fixed query can be proved correct with respect to the public key; 2)pseudorandomness. After several queries, a polynomial-time observer can not distinguish between a value that is computed by evaluating the function on his new query and a value picked at random *without the help of proof of correctness*. These distinguishing features make it useful in protocol design, as demonstrated in [21, 22].

Zero knowledge (ZK for short) proof [16, 15], a proof that reveals nothing but the validity of the assertion, is another fundamental tool in design of cryptographic protocols. In recent years, several notable notions have emerged to capture some new security concerns that arise in modern maliciously asynchronous communication environment, such as concurrent ZK[11]and universal composable ZK[7]. An other notable concept is resettable ZK (rZK) introduced by Canetti et al.[8]. The rZK formalizes security in a scenario in which the verifier is allowed to reset the prover in the middle of proof to any previous stage. From the randomness point of view, this notion is a strongest security measure. Obviously the notion of rZK is stronger than that of concurrent ZK and therefore we can not construct a constant round black-box rZK protocol in the plain model for non-trivial languages[9].

Following the above work, Barak et al. [2] initiated the study of soundness in a setting where the prover can reset the honest verifier, and showed that the public-coin constant round ZK argument of knowledge [1, 3] can be easily transformed into constant-round resettably-sound ZK argument of knowledge. Barak et al. also made a fascinating conjecture in [2]: there exist resettably-sound resettable ZK arguments for NP. Unfortunately, no progress on this conjecture has been made so far. The known results either achieved only resettable zero knowledge, such as the (non-constant round) resettable ZK proof system of [8] and the constant round public-coin bounded-resettable ZK argument system (we call it BLV's protocol) of [5], or achieved only resettable-soundness, such as resettably-sound ZK argument system of [2].

It is shown that psedurandom functions are crucial ingredients in the constructions of all known rZK or resettably-sound ZK protocols. However, the pseudorandom functions, even the stronger primitive of verifiable random functions, seem not powerful enough to tackle the simultaneous resettability problem, and this lead us to develop the new primitive—instance-dependent verifiable random functions (InstD-VRFs for short).

Motivation behind InstD-VRFs. Let's return to the resettably-sound ZK argument in [2]. We first note that if we modify this protocol in such a way that the verifier's messages (except for the first one) satisfy some kind of binding property, for example, the verifier's responses are determined by the first verifier's message and the history messages so far, then the resetting attack from the malicious verifier can be trivialized if the verifier do not reset the prover past its first message, therefore it seems the resulting protocol achieves some certain simultaneous resettability. Apparently, This can be done by plugging a verifiable random function in Barak's public-coin bounded concurrent ZK protocol: the first verifier's message includes a public key of verifiable random function along with the description of a hash, and all subsequent verifier's messages are computed by applying the verifiable random function to the history messages in a session.

At a first look, the above resulting protocol enjoys certain desirable simultaneous resettability: besides achieving a stronger ZK property than bounded resettable ZK that is achieved in the BLV protocol [5], it seems to be resettably-sound due to the pseudorandomness of the verifiable random function. Indeed, we can prove this protocol is ZK against somewhat restricted resetting verifier. However, we do not know how to prove the soundness in standard way (i.e., proving by reduction), let alone the resettably-soundness, because for the analysis of soundness to go through we typically need some freedom in verifier's responses to the same history of messages (i.e., verifier can choose different messages to reply the same history of messages)in the WI universal argument of Barak's protocol, and unfortunately, the uniqueness of the verifiable random functions precludes this possibility.

Inspired by the previous interesting instance-dependent commitment scheme [19, 20], We introduce the notion of instance-dependent verifiable random functions (InstD-VRFs), to achieve a certain simultaneous resettability. Informally, an InstD-VRF is, in some sense, a verifiable random function [23] with a special public key, which is generated via a (possibly)*interactive* protocol and contains an instance $y \in L \cap \{0,1\}^*$ for a specific NP language L, but the security requirements on such a function are relaxed: we only require the *pseudorandomness* property when $y \in L$ and only require the *uniqueness* property when $y \notin L$, instead of requiring both pseudorandomness and uniqueness to hold *simultaneously.* The reason why such functions are useful is that we use only the uniqueness to justify resettable ZK property and use only the pseudorandomness to justify resettable-soundness.

Our contributions. In this paper We introduce a notion of *instance-dependent verifiable random functions* and realize them under standard assumption. These functions yields two interesting instance-dependent argument systems, which we call key_instance-dependent resettably-sound bounded-class resettable ZK argument and resettable witness indistinguishable argument with instance-dependent weak resettably-soundness. These results extend the study of instance-dependent primitives (protocols).

The instance-dependent verifiable random functions, together with the above instance-dependent protocols, are powerful tools to tackle the simultaneous resettability conjecture. By using them, we construct the first (constant round) zero knowledge arguments for NP enjoying *certain* simultaneous resettability in the plain model. In our argument system, both the prover and the verifier are protected from some kinds of *restricted* resetting attacks: For the malicious resetting prover, we put a priori bound on the number of the first messages sent by it to each incarnation of the honest verifier, and for the malicious resetting verifier, in addition to putting the aforementioned restriction on it, We further put a priori bound on the number of incarnations of prover with which it is allowed to interact. We call this protocol the bounded-class resettable ZK argument with weak resettable-soundness. We stress our arguments assume standard (polynomial time) hardness assumptions and their resettable security for the prover (the verifier) does not rely on any restriction on the number of the total resets made by malicious verifiers (provers). This is in contrast to the BLV protocol [5] (that achieves only standard soundness), which relies on a priori bound of the total resets made by malicious verifiers and exponential hardness assumptions.

We also show that, if there exist public-coin concurrent ZK arguments for NP, the idea behind the above construction can be applied to the *unbounded* simultaneously resettable setting, and this leads to resettably-sound resetable ZK arguments for NP.

The resettable witness indistinguishable argument with instance-dependent weak resettably-soundness is a crucial and delicate component in our main constructions (in section 5), in which both the prover and the verifier use a instance-dependent verifiable random function. For the prover, the instance to be proven serves as the key_instance for its $\mathsf{InstD\text{-}VRF}$ directly. The most interesting but difficult task is to produce a key_instance for the verifier's $\mathsf{InstD\text{-}VRF}$. Our solution to this problem is to have the prover generate a NO key_instance with respect to a hard-to-decide language for the verifier and prove to the verifier that the statement to be proven is true or this key_instance is an YES instance via a resettable witness indistinguishable argument with instance-dependent weak resettably-soundness. A glaring property of this argument is that it is *argument of knowledge* when the statement to be proven (in the global system) is false, and this is the crux in the analysis of the soundness of our main argument systems presented in section 5.

Subsequent work. The instance-dependent verifiable random functions seem to have potential beyond what we demonstrate in this paper. Very recently, by using these functions in a novel way, we construct resettable witness indistinguishable argument with instance-dependent *unbounded* (in contrast to *"weak"*) resettably-soundness, and this immediately yields a (unbounded) resettably-sound bounded-class resettable ZK argument, which gets close to the simultaneous resettability conjecture.

Outline. The definition of bounded-class resettable ZK and weak resettable-soundness are presented in section 2. In section 3, we introduce the notion of instance-dependent verifiable random functions and show a construction of this primitive under standard assumption. The application of the new primitive are described in section 4, 5. In section 4 we present two interesting instance-dependent protocols that are crucial building block for our construction of bounded-class resettable ZK argument with weak resettable-soundness, and the latter argument, along with a sufficient condition for the simultaneous resettability conjecture, are presented in section 5.

2 Definitions

In this section we mainly define bounded-class resettable ZK and weak resettable-soundness, which can be viewed as intermediate notions between their full resettable analogues and bounded resettable analogues. Due to lack of space, we refer readers to [13] for some basic concepts, such as computational indistinguishability, (statistically-biding) commitment scheme, hybrid argument, and so on.

In the following We denote by $\delta \leftarrow_R \Delta$ the process of picking a random element δ from Δ, and 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 $n \geq N$, $f(n) \leq 1/q(n)$.

We follows the standard definition of zero knowledge argument in [13]. Note that for such a protocol the soundness is required to hold only against PPT adversaries.

Resettable prover and bounded-class resettable ZK Resettable ZK was introduced in [8]. In essence, it guarantees the security of a prover with fixed random tape in a scenario the verifier is allowed to run polynomial number sessions with this fixed prover.

We introduce the notion of *bounded-class* resettable ZK. We call a fixed prover strategy $P^{(i,j)} = P_{x_i,w_i,r_j}$ an incarnation. We 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. We denote a class associated with the incarnation $P^{(i,j)}$ and the verifier's first message msg with $\mathrm{Class}_{P^{(i,j)},\mathsf{msg}}$. *Note that it is possible that a class contains (unbounded) any polynomial number sessions because the verifier is allowed to reset the prover.*

Definition 1. *(Bounded-class resettable ZK argument) Let t be a polynomial. An interactive argument (P, V) for a language L is said to be t^3-bounded-class resettable ZK if for every every PPT adversary V^* there exists a PPT M so that the following two distributions are computational indistinguishable, where each distribution is indexed by a sequence of distinct common inputs $\overline{x} = x_1, \cdots, x_t \in L \cap \{0,1\}^n$ and a corresponding sequence of prover's auxiliary inputs $\overline{w} = w_1, \cdots, w_t$,*

Distribution 1 *is defined by the following random process depending on P and V.*

1. *Randomly pick and fix* t *random tapes,* $r_1, \cdots, r_t$*, resulting in* t^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* $(i,j) \in \{1, \cdots, t\} \times \{1, \cdots, t\}$.
2. *The adversary* V^* *is allowed to run polynomial many sessions with the* $P^{(i,j)}$*'s,* but for each $P^{(i,j)}$, the verifier can not reset $P^{(i,j)}$ past its first message more than $t-1$ times, that is, the number of different V^*'s first message to each incarnation $P^{(i,j)}$ is a priori bounded by t. *Under this restriction, the verifier is allowed to schedule all sessions in interleaving way:* V^* *can send arbitrary messages to each of the* $P^{i,j}$*, and obtain the responses of* $P^{(i,j)}$ *to such messages immediately.*
   This results in at most t^3 *classes in the whole interaction.*
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* $(P(\overline{w}), V^*)(\overline{x})$.

Distribution 2 *is the output of* $M(\overline{x})$.

The resetting attack performed by the restricted malicious verifier in the above definition is called *bounded-class resetting attack*. We stress there is essential difference between bounded-resettable ZK [5] and bounded-class resettable ZK: We impose no restriction on the number of the total resets (sessions) that malicious verifiers can make.

Resettable verifier and weak resettable-soundness. Following the definitions of resettablly-sound arguments in [2], we consider a *weak resetting attack*, in which a malicious prover is not allowed to reset an incarnation of the verifier past its first message more than $t-1$ times, but still can interact with *arbitrary polynomial number* of verifier's incarnations. This kind of attack corresponds to the following notion of soundness.

Definition 2. *(Weak resettably-sound argument of knowledge.) for some a-priori fixed polynomial* t*,* a weak 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 picks and fix* $poly(n)$ *random-tapes, denoted* $r_1, \cdots, r_{poly(n)}$*, for* V*, resulting in deterministic strategies* $V^{(j)}(x) = V_{x,r_j}$*,* $x \in \{0,1\}^n$ *and* $j \in \{1, \cdots, poly(n)\}$*, defined by* $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$*. We call each* $V^{(j)}(x)$ *an incarnation of* V.
2. *Taking as input* 1^n*,* P^* *is allowed to initiate* any polynomial *number sessions with the* $V^{(j)}(x)$*'s,* but the number of different P^*'s first message to each incarnation $V^{(j)}(x)$ is a priori bounded by t. *Under this restriction, the prover* P^* *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 weak resettably-sound argument of knowledge system if it satisfies:*

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. *weak Resettably-soundness: For every weak resetting attack of a PPT P^*, the probability that in some sessions the corresponding $V^{(j)}(x)$ has accepted an 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 weak 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.*

We remark that in a weak resetting attack the malicious prover is entitled to interacts with *unbounded* number of incarnations of the verifier. So we have four types of resetting attack, full resetting attack, weak resetting attack, bounded-class resetting attack and bounded resetting attack [5], each of which is more powerful than the previous one.

Resettably-sound resettable WI. Roughly speaking, witness indistinguishability arguments [12] are arguments with property that nobody can tell which witness was used to prove a statement in an interaction. Analogously, we define Resettably-sound resettable WI (cf. [2]), and its variants according to our restrictions on the number of class and/or the number of a malicious party's first messages to each incarnation of the honest party. Due to space limitations, we omit it here.

A note on terminology. Let A be a security property or a type of attack. In the rest of the paper, the notion "unbounded A" means an unrestricted version of A.

3 Instance-dependent Verifiable Random Functions

In this section we will present the formal definition of instance-dependent verifiable random functions and show how to implement it.

3.1 InstD-VRFs: Definition

As is hinted by its name, the public key for a instance-dependent verifiable function contains a instance $y \in L \cap \{0,1\}^*$ for a NP language L, and unlike the verifiable random functions, whose *pseudorandomness* and *uniqueness* are required to hold at the same time, we require an instance-dependent verifiable random function satisfies only the *pseudorandomness* when $y \in L$, and satisfies only the *uniqueness* when $y \notin L$.

Let $d, l : N \to N$ be two polynomial. Formally, an instance-dependent verifiable random function with respect to an NP language L associates with the following (interactive) algorithms:

- KGProt, the key generation protocol between two parties, the querier A and the owner of the function B, each party taking security parameter n and an

random string as input, produces a public/secret key pair (PK, SK), and PK is of form $(y, \cdot)$, where $y \in L \cap \{0,1\}^n$ is called key_instance.
- $\mathsf{F} = (f, prov)$, the function evaluator, the first component is a deterministic algorithm while the second component $prov$ is a *probabilistic* algorithm. Given a (PK, SK), on input an element $a \in \{0,1\}^{d(n)}$ (the domain of f_{SK}), it outputs a function value $b \in \{0,1\}^{l(n)}$ (the range of f_{SK}) and a proof π. That is, $\mathsf{F}_{(PK,SK)}(a) = (f_{SK}(a), prov(a, f_{SK}(a), PK, SK)) = (b, \pi)$, where $\mathsf{F}_{(PK,SK)}(\cdot) = \mathsf{F}(PK, SK, \cdot)$.
- Ver, the verification (deterministic) algorithm, on input a, b, PK and a proof π, Ver outputs 1 or 0.
- FakeF, the *fake* function evaluator. Assume y is in L. Given PK and a witness w_y for $y \in L$, for every $a \in \{0,1\}^{d(n)}$, $\mathsf{FakeF}_{(PK,w_y)}$ can validate an arbitrary false function value, i.e., for an arbitrary $b \in \{0,1\}^{l(n)}$, taking a as input, $\mathsf{FakeF}_{(PK,w_y)}$ can output $(b, prov(a, b, PK, w_y)) = (b, \pi)$ such that $\mathsf{Ver}(a, b, PK, \pi) = 1$.

The property of the fake function evaluator guarantees for a function $h : \{0,1\}^{d(n)} \to \{0,1\}^{l(n)}$ that deviates arbitrarily from the function f_{SK} specified by the secret key, We can run the algorithm $prov$ using w_y to produce a valid proof of correctness for the function value $h(a)$. We define the following two useful fake functions:

- $\mathsf{FakeF}_{(PK,w_y,s)}(a) \triangleq (f_s(a), prov(a, f_s(a), PK, w_y))$, where $f_s : \{0,1\}^{d(n)} \to \{0,1\}^{l(n)}$ is an arbitrarily (independent of f_{SK}) pseudorandom function.
- $\mathsf{FakeF}_{(PK,w_y,h)}(a) \triangleq (h(a), prov(a, h(a), PK, w_y))$, where $h : \{0,1\}^{d(n)} \to \{0,1\}^{l(n)}$ is an arbitrarily (truly) random function.

Note that the algorithm $prov$ in $\mathsf{FakeF}_{(PK,w_y)}$ produces a valid proof without knowledge of the secret key SK, or the seed s (the description) of the function f_s (h) plugged in.

We say $\mathsf{F}_{(PK,SK)}(\cdot)$ is a *instance-dependent verifiable function* if it satisfies the following conditions:

1. *Provability.* If $(b, \pi) = \mathsf{F}_{(PK,SK)}(a)$, then $\mathsf{Ver}(a, b, PK, \pi) = 1$
2. *Uniqueness on NO key_instance.* If $y \notin L$, then except with a negligible probability, there exist no values $(a, b_1, b_2, PK, \pi_1, \pi_2)$ such that $\mathsf{Ver}(a, b_1, PK, \pi_1) = \mathsf{Ver}(a, b_2, PK, \pi_2) = 1$
3. *Pseudorandomness on YES key_instance.* If $y \in L$ and w_y is a witness for $y \in L$, then for every PPT oracle machine M, every polynomial p, and all sufficient large $n's$,

$$|[Pr[M^{F_{(PK,SK)}}(1^n) = 1] - Pr[M^{\mathsf{FakeF}_{(PK,w_y,h)}}(1^n) = 1 : h \leftarrow_R H_n]| < 1/p(n)$$

where H_n is the ensemble of all functions mapping $d(n)$-bit-long strings to $l(n)$-bit-long strings.

Remarks:

On key generation protocol. In contrast to the verifiable random functions [23] whose keys are generated by the function owner alone, our instance-dependent verifiable random functions allows interaction between the querier and the owner of the function in the key generation process. Note also that in above definition we do not make any requirement on the key generation protocol. This will allow us to design different key generation protocols for different purposes, see details in section 4.2 and section 5.

On pseudorandomness when $y \in L$. We remark that in the testing experiment, M obtains the function value *along with its proof of correctness* for every string that the oracle machine M queried. This is different from the testing experiment used to demonstrate the pseudorandomness of verifiable random functions, in which providing the proof of correctness along with the function value of the last query for judgement to the testing machine will trivialize this test because of the uniqueness of the verifiable random functions.

3.2 InstD-VRFs: Constructions

We begin with an informal description of our construction. The querier A and the function owner B execute a 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 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. The public key of this instance-dependent verifiable random function consists of the instance y, the commitment and the setup information for the WI proof, the secret key is the decommitment.

For our applications, we require that the proof in use satisfies both resettable-soundness and resettable WI. To this end, the 2-round ZAPs introduced in [10] and the non-interactive ZAPs suggested in [17] are good candidates. Here we adopt 2-round ZAPs just for the purpose of basing our results on more general assumptions.

In the key generation protocol KGProt, the way to produce an key_instance for the owner of the function is a subtle problem and may vary depending on specific applications. In our applications there are two approaches to do this: for the function owned by the prover, the key_instance in its public key is the instance to be proven (therefore the honest prover supposedly knows the corresponding witness w_y); for the function owned by the verifier, the key_instance will be generated by the prover and the prover gives a special WI argument in which it proves the statement to be proven is true or the key_instance generated by itself is an YES instance(therefore the instance generated by an honest prover is a NO instance, and this guarantees the uniqueness property of this function). See details in section 5.

Bearing the above in mind, we omit the formal description of KGProt here for greater flexibility. Now, we simply assume that the key_instance y has been generated already. The rest components of the key pair are created in following way: the function owner B picks a pseudorandom functions f_{s_0} from the ensemble $\{f_s : \{0,1\}^{d(n)} \to \{0,1\}^{l(n)}\}_{s\in\{0,1\}^n}$, and commits to the seed s_0 using a statistically binding commitment scheme Com, let $c = Com(s_0, r)$ (r is the randomness required by the commitment scheme). The querier A selects a random string ρ as the first round message of ZAP and send it to B. On received ρ, the function owner B publish the public key $PK = (y, c, \rho)$ and keep $SK = (s_0, r)$ as the secret key.

Given (PK, SK), on input $a \in \{0,1\}^{d(n)}$, $\mathsf{F}_{(PK,SK)}$ returns a function value b and the second round message of ZAP π (the proof) in which it proves that there exist (s_0, r) such that $f_{s_0}(a) = b$ and $c = Com(s_0, r)$ or $y \in L$. i.e., $\mathsf{F}_{(PK,SK)}(a) = (f_{s_0}(a), prov(a, f_{s_0}(a), PK, SK)) = (b, \pi)$. Here we can view the probabilistic algorithm *prov* as the prover in a ZAP system.

In the fake function evaluator $\mathsf{FakeF}_{(PK,w_y)}$, by using the witness w_y to the YES instance y, the algorithm *prov* can always generate the valid proof of correctness regardless of whether the function value is correct or not.

Theorem 1. If there exist trapdoor permutations, there exist instance-dependent verifiable random functions.

Proof. We prove that The function evaluator $\mathsf{F}_{(PK,SK)}$ described above is an instance-dependent verifiable random function. Note that the statistically-binding commitment scheme and pseudorandom functions can be constructed based on one-way fucntions [24, 18], and ZAPs assumes only trapdoor permutations.

The Provability is straightforward. *Uniqueness on NO key_instance* follows immediately from the statistically-binding property of Com and the soundness of ZAPs.

We prove the *Pseudorandomness on YES key_instance* using hybrid arguments. For every PPT oracle machine M, we consider the following sequences of hybrids, in each hybrid M makes a polynomial number of queries to a function that is slightly different from the one in previous hybrid. We complete the proof by showing M distinguishes each hybrid from its neighbor with at most a negligible probability.

Hybrid 0 M queries $\mathsf{F}_{(PK,SK)}$.

Hybrid 1 M queries $\mathsf{F}'_{(PK,SK)}$, where $\mathsf{F}'_{(PK,SK)}(\cdot) = (f_{s_0}(\cdot), prov(\cdot, f_{s_0}(\cdot), PK, w_y))$. That is, $\mathsf{F}'_{(PK,SK)}$ behaviors as the same as $\mathsf{F}_{(PK,SK)}$ except it produces the proof of correctness using the witness w_y to $y \in L$. The fact that $\mathsf{F}_{(PK,SK)}$ and $\mathsf{F}'_{(PK,SK)}$ are indistinguishable follows immediately from the witness indistinguishability of ZAPs.

Hybrid 2 M queries $\mathsf{FakeF}_{(PK,w_y,s)}$, where the *pseudorandom* function seed s is selected at random. We claim that M cannot distinguish $\mathsf{F}'_{(PK,SK)}$ from $\mathsf{FakeF}_{(PK,w_y,s)}$ with non-negligible probability. Assume otherwise, we construct a non-uniform algorithm D to break the hiding property of the statistically binding commitment Com. We give the detailed proof later.

Hybrid 3 M queries $\mathsf{FakeF}_{(PK,w_y,h)}$, where h is a *truly random* function. Note that the proof of correctness has nothing to do with the seed s or the description of h, so if M distinguishes $\mathsf{FakeF}_{(PK,w_y,s)}$ from $\mathsf{FakeF}_{(PK,w_y,h)}$, it distinguishes a pseudorandom function form a truly random function.

Now we give the description of algorithm D to prove the claim in Hybrid 2. D runs as follows. It takes s_0,s_1 and w_y[1] as input. On received the target commitment c' that is the commitment to s_0 or s_1, D uses $PK = (y, c', \rho)$ as public key, and for any query a made by M, it returns $f_{s_0}(a)$ and $prov(a, f_{s_0}(a), PK, w_y))$ (Note that D always uses f_{s_0} to compute the function value, then when $c' = Com(s_0)$, D performs as $\mathsf{F}'_{(PK,SK)}$; when $c' = Com(s_1)$, D performs as $\mathsf{FakeF}_{(PK,w_y,s_0)}$). At end, if M outputs $b \in \{0,1\}$, D output $1-b$. We show D breaks the hiding property of Com. We assume for some polynomial p,

$$|[Pr[M^{\mathsf{F}'_{(PK,SK)}}(1^n) = 1] - Pr[M^{\mathsf{FakeF}_{(PK,w_y,s)}}(1^n) = 1 : s \leftarrow_R \{0,1\}^n]| > 1/p(n)$$

Then we have

$$\begin{aligned}
&|[Pr[D \text{ outputs } 1 | c' = Com(s_1)] - Pr[D \text{ outputs } 1 | c' = Com(s_0)]\\
=&|[Pr[M^{\mathsf{FakeF}_{(PK,w_y,s_0)}}(1^n) = 0] - Pr[M^{\mathsf{F}'_{(PK,SK)}}(1^n) = 0]|\\
=&|[Pr[M^{\mathsf{F}'_{(PK,SK)}}(1^n) = 1] - Pr[M^{\mathsf{FakeF}_{(PK,w_y,s_0)}}(1^n) = 1]|\\
>&1/p(n)
\end{aligned}$$

A note on input length. We remark that an $\mathsf{InstD\text{-}VRF}$ $\mathsf{F}_{(PK,SK)}$ with domain $\{0,1\}^{d(n)}$ can also be applied to inputs of length shorter than $d(n)$ by simply encoding the shorter inputs into the ones of desired length (cf. [13]) and using a prefix of ρ with suitable length as the first round message of a ZAP for the proof of correctness.

4 Two Instance-dependent Protocols

With the instance-dependent verifiable random functions we developed, We are ready to construct two interesting instance-dependent protocols, which we call key_instance-dependent resettabley-sound bounded-class resettable ZK argument (KInstD rs-rZK argument) and resettable WI argument with instance-dependent weak resettable-soundness (InstD rs-rWI argument). Though these protocols do not even satisfy ZK (WI) and (knowledge) soundness at the same time, they are crucial tools for our main constructions presented in next sections.

4.1 Key_instance-dependent Resettabley-sound Bounded-class Resettable ZK Arguments for NP

We first show how to transform a public-coin bounded concurrent ZK argument into a key_instance-dependent resettabley-sound bounded-class resettable

[1] In case y is generated by the querier in the key generation protocol, it seems that y must be generated before the commitment c is seen by the querier, otherwise, our non-uniform algorithm D does not work because it needs to take a fixed advice (i.e.,the witness to y) in advance (before seeing the target commitment) in breaking the hiding property of the commitment. However, in our applications, we do not need comply with this order. To enable the above analysis, we require the querier give a special argument of knowledge of the witness to y which is generated by itself.

ZK argument by equipping the verifier in the former system with an instance-dependent verifiable random function. Similar to any other instance-dependent primitive, the key_instance-dependent resettabley-sound bounded-class resettable ZK argument satisfies only the resettable-soundness when the key_instance is a YES instance, and satisfies only the bounded-class resettable ZK when the key_instance is a NO instance.

As showed in [2], we can transform a constant-round public-coin bounded concurrent ZK argument $(P_{\rm B}, V_{\rm B})$ into a constant round resettably-sound bounded concurrent ZK argument $(P_{\rm R}, V_{\rm R})$ by simply equipping $V_{\rm R}$ with a pseudorandom function and letting $V_{\rm R}$ emulate $V_{\rm B}$ except that it generate the current round message by applying a pseudorandom function to the transcript so far. With the argument $(P_{\rm R}, V_{\rm R})$, We construct a key_instance-dependent resettably-sound bounded-class resettable ZK argument (KInstD rs-rZK argument, for short) (P, V) as follows. The prover P and the verifier V first execute a key generation protocol KGProt aimed at setting up a key pair (PK, SK) of an instance-dependent verifiable random function $\mathsf{F}_{(PK,SK)} = (f_{s_0}, prov)$ with respect to a hard language L' (the choice of language L' see section 5) for V, and then they execute the protocol $(P_{\rm R}, V_{\rm R})$ in following way: 1) In each P's step, P checks whether the message sent by V is computed correctly, if so, it replies according to the instruction of $P_{\rm R}$; 2) V feeds $V_{\rm R}$ with the randomness s_0, and in each V's step, V generates its message by running $V_{\rm R}$ and using the algorithm $prov$ in $\mathsf{F}_{(PK,SK)} = (f_{s_0}, prov)$ to give a proof of correctness for the output by $V_{\rm R}$. Note that $V_{\rm R}$ always generates its message by applying f_{s_0} to the history produced by $P_{\rm R}$ and $V_{\rm R}$, so each V's message can be viewed as the output by $\mathsf{F}_{(PK,SK)} = (f_{s_0}, prov)$ on the *the transcript of the underlying protocol* $(P_{\rm R}, V_{\rm R})$ so far. See Fig.1 for the formal description.

We assume the transcript size of an execution of the resettably-sound t^3-bounded concurrent ZK argument $(P_{\rm R}, V_{\rm R})$ is bounded by a polynomial d, and assume the longest message sent by $V_{\rm R}$ is t^3n^3. Without of loss generality, We assume all verifier's messages are of equal length.

Theorem 2. The KInstD rs-rZK argument (P, V) depicted in Fig.1 satisfies following conditions:

1. Unbounded resettable-soundness when $y \in L'$: for any $x \notin L$, if all key_instance y's generated by an incarnation $V^{(j)}(x) = V_{x,r_{v_j}}$ are in L, then for any PPT P^* mounting unbounded resetting attack, the probability that $V^{(j)}(x)$ accept in some session is negligible.
2. t^3-Bounded-class resettable ZK when $y \notin L'$: For all PPT V^* mounting bounded-class resetting attack, if $y \notin L'$ holds for all sessions, then there exists a PPT M satisfying the requirement of Definition 1.

Intuitively, the property of resettable-soundness when $y \in L'$ follows from the fact that once the verifier has the witness, it can send arbitrary messages to a same history of a session without being detected by the prover, so we can reduce the soundness of the KInstD rs-rZK argument (P, V) to the underlying protocol $(P_{\rm R}, V_{\rm R})$. On the other hand, if all y's are NO instance, then the first

KInstD rs-rZK Argument (P, V)

Common input: $x \in L$ ($|x| = n$)
The Prover's private input: a witness w for $x \in L$.
Prover's randomness: r_p, a seed of a pseudorandom function f_{r_p}
Verifier's randomness: r_v, a seed of a pseudorandom function f_{r_v}

Phase 1: the key generation protocol KGProt

$V \to P$ V sets $(r_v^1, r_v^2) = f_{r_v}(x)$, selects $f_{s_0} \leftarrow_R \{f_s : \{0,1\}^{\leq d(n)} \to \{0,1\}^{t^3 n^3}\}_{s \in \{0,1\}^n}$ and $r_0 \leftarrow_R \in \{0,1\}^n$ using randomness r_v^1, computes $c_0 = Com(s_0, r_0)$ using the statistically-binding commitment scheme Com and generates an instance $y \in L' \cap \{0,1\}^n$, stores $SK = (s_0, r_0)$. Sends c_0, y;

$P \to V$ P sets $(r_p^1, r_p^2) = f_{r_p}(x, c_0, y)$, selects the first message ρ for a ZAP using randomness r_p^1. at random. At the end of this step, the $\mathsf{InstD\text{-}VRF}$'s key pair $(PK, SK) = ((y, c_0, \rho), (s_0, r_0))$ is set up for V.
Sends ρ;

Phase 2: the *Modified* resettably-sound t^3-bounded concurrent ZK argument

$V \Leftrightarrow P$ Let $(P_{\rm R}, V_{\rm R})$ be the resettably-sound t^3-bounded concurrent ZK argument. P writes r_p^2 on $P_{\rm R}$'s random tape. V writes s_0 on $V_{\rm R}$'s random tape (the description of f_{s_0}). P and V perform the following strategy.

V**'s Strategy** In each V's step, V runs $V_{\rm R}(s_0, \cdot)$ on input $hist$, where $hist$ is the history (including the common input) produced by $P_{\rm R}$ and $V_{\rm R}$ (*not the history produced by P and V*) so far. V obtains $m_v = V_{\rm R}(s_0, hist)$, and computes $\pi = prov(hist, m_v, PK, SK)$ using randomness $f_{r_v^2}(hist)$, then V sends (m_v, π) to P.
Note that $(m_v, \pi) = (f_{s_0}(hist), prov(hist, m_v, PK, SK)) = \mathsf{F}_{(PK,SK)}(hist)$ ($V_{\rm R}$ always generates m_v by applying f_{s_0} to the history so far).

P**'s Strategy** In each P's step, P checks whether the message (m_v, π) sent by V is correct by using the algorithm Ver associated with $\mathsf{F}_{(PK,SK)}$, if not, aborts; if so, runs $P_{\rm R}(r_p^2, \cdot)$ on input $(hist, m_v)$, here $hist$ is the history produced by $P_{\rm R}$ and $V_{\rm R}$ before the message m_v was sent by V. P sends $m_p = P_{\rm R}(r_p^2, hist, m_v)$ to V.

V**'s Decision** V accepts if only if $V_{\rm R}$ accepts the transcript that generated by $P_{\rm R}$ and $V_{\rm R}$.

Fig. 1. The key_instance-dependent resettably-sound bounded-class resettable ZK argument for a NP language L.

verifier's message essentially determines the verifier's behavior, this will make our protocol enjoy bounded-class resettable ZK. The actual proof is omitted here, and will be found in the full version of this paper.

4.2 The Resettable WI Arguments with Instance-dependent Weak Resettable-soundness

In this subsection, we construct resettable WI arguments with instance-dependent weak resettable-soundness (InstD rs-rWI argument, for short), in which the prover proves that one of the two instances x_0 and x_1 is in the language L. Though there are resetably-sound resettable WI arguments for NP such as ZAPs, which achieves more stronger security than our InstD rs-rWI arguments, however, the InstD rs-rWI arguments have a distinguishing property that ZAPs do not enjoy: they are *arguments of knowledge* on some special instances. This property is crucial for the analysis of soundness of our main construction presented in next section.

We assume that there are 3 round public-coin WI arguments of knowledge for all NP languages. Let (a, e, z) is the three messages exchanged in a session. Furthermore, we assume these arguments have the following property: it is easy to extract the witness for the statement from two different transcripts (a, e, z) and (a, e', z') when $e \neq e'$. To this end, the parallelized version of Blum's proof of knowledge for Hamiltonian Cycle is a good candidate, which assumes one-way permutations exist.

Our construction is inspired by the protocol 6.2 in [2], in which the verifier first commits to a seed of a pseudorandom function and generates the query e by applying this function to the first round message a. The important deviation is that the verifier in our system uses a KInstD rs-rZK argument to prove the query e is computed correctly. In the KInstD rs-rZK argument, one instance, say x_0, serves as the key_instance for a InstD-VRF used by the verifier (the prover in the global system). The formal description appears in Fig.2.

Theorem 3. The InstD rs-rWI argument $(P_{\mathtt{W}}, V_{\mathtt{W}})$ depicted in Fig.2 satisfies following properties:

1. *Unbounded* Resettable witness indistinguishability: For any PPT $V_{\mathtt{W}}^*$ mounting *unbounded* resetting attack, the distribution $(P_{\mathtt{W}}(\overline{w_0}), V_{\mathtt{W}}^*)(\overline{x})$ is computationally indistinguishable from $(P_{\mathtt{W}}(\overline{w_1}), V_{\mathtt{W}}^*)(\overline{x})$, where $\overline{x} = x^1, \cdots, x^{poly(n)}$, $x^i = (x_0^i, x_1^i)$, $\overline{w_b} = w_b^1, \cdots, w_b^{poly(n)}$ such that $(x_b^i, w_b^i) \in R_L$, $i = 1, \cdots, poly(n)$, $b = 0, 1$.
2. Weak resettably-sound argument of knowledge property when $x_0 \notin L$: For every PPT $P_{\mathtt{W}}^*$ mounting *weak* resetting attack, if $P_{\mathtt{W}}^*$ convinces $V_{\mathtt{W}}$ on statement (x_0, x_1) such that $x_0 \notin L$ with probability p in a session, then there exists a PPT machine E, upon input the description of $P_{\mathtt{W}}^*$, outputs a witness for the instance x_1 with probability negligibly close to p.

The *Unbounded* Resettable witness indistinguishability follows from the fact that the underlying KInstD rs-rZK argument satisfies resettably-soundness when $x_0 \in$

L. For the Weak resettably-sound argument of knowledge property when $x_0 \notin L$, We can construct a extractor E to justify it. Assume $P^*_{\mathtt{W}}$ convinces an incarnation $V^j_{\mathtt{W}}(x)$ on statement (x_0, x_1) such as $x_0 \notin L$ with high probability in a session. The extractor E first plays the role of $V^j_{\mathtt{W}}(x)$ and get an accepting transcript

InstD rs-rWI Argument $(P_{\mathtt{W}}, V_{\mathtt{W}})$

Common input: two instance $x_0, x_1 \in L$, a security parameter n.
The Prover's private input: the witness w such that $(x_0, w) \in R_L$ or $(x_1, w) \in R_L$.
Prover's randomness: r_p, a seed of a pseudorandom function f_{r_p}
Verifier's randomness: r_v, a seed of a pseudorandom function f_{r_v}

$V_{\mathtt{W}} \to P_{\mathtt{W}}$	$V_{\mathtt{W}}$ sets $(r_v^1, r_v^2) = f_{r_v}(x_0, x_1)$. Using the randomness r_v^1, $V_{\mathtt{W}}$ selects a pseudorandom function $f_s : \{0,1\}^{\leq poly(n)} \to \{0,1\}^{\|e\|}$ and r, computes $c = Com(s, r)$ using a statistically-binding commitment scheme Com. Sends c;
$P_{\mathtt{W}} \to V_{\mathtt{W}}$	$P_{\mathtt{W}}$ sets $(r_p^1, r_p^2) = f_{r_p}(x_0, x_1, c)$. Using the randomness r_p^1 , $P_{\mathtt{W}}$ invokes the 3 round WI argument in which it proves $x_0 \in L$ or $x_1 \in L$, produces the first message a of this protocol . using the randomness r_p^2, $P_{\mathtt{W}}$ invokes a KInstD rs-rZK argument (in which $P_{\mathtt{W}}$ plays the role of verifier), produces the first message c_0 (i.e., the commitment to the description of pseudorandom function) and uses x_0 as the key_instance. Send a,c_0;
$V_{\mathtt{W}} \to P_{\mathtt{W}}$	$V_{\mathtt{W}}$ computes $e = f_s(x_0, x_1, c, a, c_0)$. Using the randomness r_v^2, $V_{\mathtt{W}}$ and selects the first message ρ for a ZAP according to the KInstD rs-rZK argument. Sends e, ρ;
$P_{\mathtt{W}} \Leftrightarrow V_{\mathtt{W}}$	$P_{\mathtt{W}}$ and $V_{\mathtt{W}}$ continue to run the KInstD rs-rZK argument in which $V_{\mathtt{W}}$ proves there exist s, r such that $e = f_s(x_0, x_1, c, a, c_0)$ and $c = Com(s, r)$. The public key for $P_{\mathtt{W}}$'s InstD-VRF is $PK = (x_0, c_0, \rho)$ and the corresponding secret key is the decommitment to c_0.
$P_{\mathtt{W}} \to V_{\mathtt{W}}$	Sends the answer z to the query e according to the 3 round WI argument if the above transcript is accepting.
$V_{\mathtt{W}}$**'s Decision**	$V_{\mathtt{W}}$ accepts if only if the transcript (a, e, z) is accepting.

Fig. 2. The resettable WI arguments with instance-dependent weak resettable-soundness.

(a, e, z) of the underlying 3-round WI argument, and then rewinds $P^*_{\mathtt{W}}$ to the point that the message that contains a was first sent by $P^*_{\mathtt{W}}$, and then sends another challenge e' and runs the simulator associated with the KInstD rs-rZK argument to prove that e' is correctly computed. At the end E will receive another accepting transcript (a, e', z') with probability close to the probability that $P^*_{\mathtt{W}}$ convinces $V^j_{\mathtt{W}}(x)$, and this allows E to compute a witness for x_1 (x_0 is assumed to be NO instance). Due to space limitations, the formal analysis of this extractor are omitted here and will appear the full version of this paper.

We stress that the InstD rs-rWI argument achieves *weak* resettably-sound argument of knowledge property when $x_0 \notin L$, rather than *bounded-class* resettably-sound argument of knowledge property when $x_0 \notin L$. There are two reasons for this: 1) it is sufficient to consider only one incarnation of the verifier in the analysis of soundness; 2) the simulation performed by E will be run successful (due to the bounded resettable zero knowledge property when $x_0 \notin L$ of the underlying KInstD rs-rZK argument, note x_0 is the key_instance for $P^*_{\mathtt{W}}$'s InstD-VRF).

However, we failed to achieve *unbounded* resettable-soundness. This is because, for justifying the extraction, we need to simulate all proofs given by $V^j_{\mathtt{W}}(x)$, not just those proofs given by $V^j_{\mathtt{W}}(x)$ with the first $P^*_{\mathtt{W}}$'s ($P^*_{\mathtt{W}}$ plays the role of the verifier in the underlying KInstD rs-rZK argument) first message (a, c_0) (therefore, we need to put a priori bound on the number of the $P^*_{\mathtt{W}}$'s first messages). Recently, we overcome this obstacle by using the the instance-dependent verifiable random functions in a novel way.

5 Transforming Public-coin (Bounded) Concurrent ZK Arguments to (Bounded-class) Resettable ZK Arguments with (Weak) Resettable-soundness

We now show how to transform a public-coin bounded concurrent ZK argument into a bounded-class resettable ZK argument with weak resettable-soundness, using the two instance-dependent protocols developed in last section. Furthermore, if public-coin (unbounded) concurrent ZK argument exists, the same transformation yields a (unbounded) resettably-sound resettable ZK argument immediately.

We obtain a bounded-class resettable ZK argument with weak resettable-soundness from a public-coin bounded concurrent ZK argument in the following way: we first transform the public–coin bounded concurrent ZK argument into a key_instance-dependent resettably-sound bounded-class resettable ZK argument, then we modify the resulting protocol in such a way taht, in the key generation protocol, instead of having the verifier generates a key_instance itself, we have the prover generates a NO instance with respect to some hard-to-decide language L' as the key_instance for the verifier's InstD-VRF and gives a proof that the statement $x \in L$ to be proven is true or this key_instance is a YES instance via a resettable WI argument with instance-dependent weak resettable-soundness in which the prover uses x as the key_instance for its own InstD-VRF, The second phase of the key_instance-dependent resettably-sound bounded-class resettable ZK argument remains unchanged. For the language L', we choose the one defined by a pseudorandom generator [6, 25]: $L' = \{y|\exists\delta, y = G(\delta), |y| = 2n, |\delta| = n\}$, where $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ is a pseudorandom generator. This protocol is formally depicted in Fig.3.

The key idea behind this protocol is that for an honest prover, it always generates NO instance as key_instance of the verifier's InstD-VRF, and for a false statement $x \notin L$, the cheating prover must generate YES instance due to the weak resettably-sound argument of knowledge property when $x \notin L$ of the InstD rs-rWI argument used in the key generation protocol. Note that the common input x itself serves as the key_instance for the prover's InstD-VRF in the InstD rs-rWI argument.

Theorem 4. If there exist trapdoor permutations and collision-resistant hash functions, there exist bounded-class resettable ZK arguments with weak resettable-soundness for all NP languages.

We prove this theorem by showing the wrs-brZK argument described in Fig.3 is a bounded-class resettable ZK argument with weak resettable-soundness for any NP language L. For the complexity assumption, we note that one-way functions are sufficient for the pseudorandom generator, so, our protocol assumes trapdoor permutations and collision-resistant hash functions (for Barak's protocol).
Proof. **Completeness** is straightforward.

Weak resettable-soundness. Assume a PPT $\mathbf{P}^*$ mounting a *weak* resetting attack, convinces an incarnation $\mathbf{V}^j(x)$ on a false statement $x \notin L$ with non-negligible probability p. By the weak resettably-sound argument of knowledge property when $x \notin L$ (note that x serves as the key_instance of $\mathbf{P}^*$'s InstD-VRF) of the InstD rs-rWI argument in the key generation protocol, we have with probability essentially close to p there exists $\exists \delta, |\delta| = n$ such that $y = G(\delta)$ (assume y is the instance generated by $\mathbf{P}^*$), and use the extractor associated with the InstD rs-rWI argument, we will extract the witness δ. Furthermore, note that y serves as the key_instance of the verifier's InstD-VRF in phase 2, then using this witness and the strategy $\mathbf{P}^*$, we can break the resettable-soundness of the underlying resettably-sound bounded-concurrent ZK argument in phase 2 in a way similar to the analysis of soundness for the KInstD rs-rZK argument, which leads to a contradiction.

Bounded-class resettable ZK. This property follows from the next lemma.

Lemma 1. Let $(P_{\mathrm{R}}, V_{\mathrm{R}})$ be a resettably-sound t^3-bounded concurrent ZK argument system, and $(\mathbf{P}, \mathbf{V})$ be the wrs-brZK argument transformed from $(P_{\mathrm{R}}, V_{\mathrm{R}})$. Then for every PPT $\mathbf{V}^*$ bounded-class resettable model, there exists a PPT V_{R}^* in the bounded concurrent model such that $(P_{\mathrm{R}}(\overline{w}), V_{\mathrm{R}}^*)(\overline{x})$ is computationally indistinguishable from $(\mathbf{P}(\overline{w}), \mathbf{V}^*)(\overline{x})$, where $\overline{x} = x_1, \cdots, x_t \in L$, $\overline{w} = w_1, \cdots, w_t$ such that $(x_i, w_i) \in R_L$, $i = 1, \cdots, t$.

Proof. We construct V_{R}^* in bounded concurrent model using the following strategy to handle $\mathbf{V}^*$'s message.

1. $\mathbf{V}^*$ sends a new first message msg to $\mathbf{P}^{(i,j)}$: Assume this is the kth new first message to $\mathbf{P}^{(i,j)}$ $(1 \leq i, j, k \leq t)$. V_{R}^* chooses δ randomly, generates $y = G(\delta)$ itself, and stores (y, δ). It acts as honest prover but uses δ as the witness to execute the InstD rs-rWI argument, and forwards a message to $\mathbf{V}^*$. Furthermore, V_{R}^* maintains a table, in which the row with index (i, j, k) contains those messages belonging to the class $\mathrm{Class}_{\mathbf{P}^{(i,j)}, \mathsf{msg}_k}$.

2. $\mathbf{V}^*$ repeats a first message msg to $\mathbf{P}^{(i,j)}$. Assume msg equals msg_k that V_{R}^* received before. V_{R}^* retrieves its response to this message from $\mathrm{row}_{(i,j,k)}$ in its table and forwards it to $\mathbf{V}^*$.

wrs-brZK Argument $(\mathbf{P}, \mathbf{V})$

Common input: $x \in L$ ($|x| = n$).
The Prover's private input: the witness w such that $(x, w) \in R_L$.
Prover's randomness: r_p, a seed of a pseudorandom function f_{r_p}
Verifier's randomness: r_v, a seed of a pseudorandom function f_{r_v}

Phase 1: the key generation protocol KGProt

$\mathbf{V} \rightarrow \mathbf{P}$ $\mathbf{V}$ sets $(r_v^1, r_v^2, r_v^3) = f_{r_v}(x)$, selects f_{s_0} and $r_0 \in \{0,1\}^n$ using randomness r_v^1, computes $c_0 = Com(s_0, r_0)$ using the statistically-binding commitment scheme Com, and stores $SK = (s_0, r_0)$.
Sends c_0;

$\mathbf{P} \Leftrightarrow \mathbf{V}$ $\mathbf{P}$ sets $(r_p^1, r_p^2, r_p^3, r_p^4) = f_{r_p}(x, c_0)$ and generates a random string y ($|y| = 2n$) using randomness r_p^1.
$\mathbf{P}$ and $\mathbf{V}$ run a InstD rs-rWI argument in which $\mathbf{P}$ proves that $x \in L$ or $\exists \delta, |\delta| = n$ such that $y = G(\delta)$. In the underlying the KInstD rs-rZK argument used in this InstD rs-rWI argument, x serves as the $\mathsf{InstD\text{-}VRF}$'s (owned by the prover in global system) key_instance, the randomness used by $\mathbf{P}$ is r_p^2 and the randomness used by $\mathbf{V}$ is r_v^2.

$\mathbf{P} \rightarrow \mathbf{V}$ $\mathbf{P}$ selects the first message ρ for a ZAP using randomness r_p^3. At the end of this step, the $\mathsf{InstD\text{-}VRF}$'s key pair $(PK, SK) = ((y, c_0, \rho), (s_0, r_0))$ is set up for $\mathbf{V}$.
Sends ρ;

Phase 2: the *Modified* resettably-sound bounded concurrent ZK argument

$\mathbf{V} \Leftrightarrow \mathbf{P}$ $\mathbf{P}$ and $\mathbf{V}$ follows the same strategy as described in the phase 2 in the KInstD rs-rZK argument (see Fig.1), in which the verifier uses an $\mathsf{InstD\text{-}VRF}$ described by $(PK, SK) = ((y, c_0, \rho), (s_0, r_0))$. In this phase, $\mathbf{P}$ uses randomness r_p^4 and $\mathbf{V}$ uses randomness r_v^3.

Fig. 3. The bounded-class resettable ZK argument with weak resettable-soundness for a NP language L.

3. $\mathbf{V}^*$ sends a *valid* non-first message belonging to the key generation protocol KGProt to $\mathbf{P}^{(i,j)}$. V_{R}^* produces the response to this message[2] according to the key generation protocol KGProt as the honest prover but in the execution of InstD rs-rWI argument it uses the δ as the witness. V_{R}^* stores this response, and forwards it to $\mathbf{V}^*$.

[2] Without loss of generality, we assume each message sent by V^* is prepended with a session ID.

Note that $\mathbf{V}^*$ is free in the key generation protocol, Thus a class recorded by V_{R}^* may contain many different sessions due to $\mathbf{V}^*$'s resetting in phase 1.

4. $\mathbf{V}^*$ sends a *invalid* message belonging to the the key generation protocol KGProt to $\mathbf{P}^{(i,j)}$. V_{R}^* sends an abort message to $\mathbf{V}^*$ to end this session.
5. $\mathbf{V}^*$ sends a *valid* message belonging to the *Modified* resettably-sound bounded concurrent zero knowledge argument (i.e., a message sent in phase 2) to $\mathbf{P}^{(i,j)}$. Assume that this message is the lth round message belonging to $\mathrm{Class}_{\mathbf{P}^{(i,j)},\mathsf{msg}_k}$. Parse this message into (m_l, π_l). We distinguish three cases:
   Case 1. The lth round message was sent by $\mathbf{V}^*$ before in some previous session in this class and the current message m_l does not equal the lth round message m'_l recorded in the row $\mathrm{row}_{(i,j,k)}$. In this case, V_{R}^* **terminates**.
   Case 2. The lth round message was never sent by $\mathbf{V}^*$ before in this class. In this case, V_{R}^* forwards m_l to the incarnation P_{R}^{i,j_k}, stores m_l and P_{R}^{i,j_k}'s response in $\mathrm{row}_{(i,j,k)}$ and forwards it to $\mathbf{V}^*$.
   Case 3. The lth round message was sent by $\mathbf{V}^*$ before in some previous session in this class and the current message m_l equals the lth round message m'_l recorded in the row $\mathrm{row}_{(i,j,k)}$. In this case, V_{R}^* retrieves the P_{R}^{i,j_k}'s response to this message in the $\mathrm{row}_{(i,j,k)}$ and forwards it to $\mathbf{V}^*$. We stress that in this case V_{R}^* does not interact with any incarnation of P_{R}.

   Observe that for two different first verifier $\mathbf{V}^*$'s messages $\mathsf{msg}_m \neq \mathsf{msg}_n$ to the same incarnation $\mathbf{P}^{(i,j)}$, V_{R}^* initiates two independent incarnations P_{R}^{i,j_m} and P_{R}^{i,j_n} to generate the $\mathbf{V}^*$'s view, and this strategy makes V_{R}^* look like the real incarnation $\mathbf{P}^{(i,j)}$.
6. $\mathbf{V}^*$ sends a *invalid* message belonging to the *Modified* resettably-sound bounded concurrent zero knowledge argument to $\mathbf{P}^{(i,j)}$. V_{R}^* sends an abort message to $\mathbf{V}^*$ to end this session.
7. $\mathbf{V}^*$ terminates. Without loss of generality, $\mathbf{V}^*$ outputs its view in the whole interaction. V_{R}^* outputs what $\mathbf{V}^*$ outputs.

It is easy to see that the strategy V_{R}^* works in bounded concurrent models if $\mathbf{V}^*$ works in bounded-class resettable model. Since V_{R}^* runs only one session with each P_{R}^{i,j_k}'s, we can identify each class $\mathrm{Class}_{P^{(i,j)},\mathsf{msg}_k}$ with the single session of V_{R}^* with P_{R}^{i,j_k} (though a class contains many different sessions, but all those sessions have the same tail, i.e., the transcript produced by P_{R}^{i,j_k} and V_{R}^* in phase 2).

Note that in sessions having different first verifier's message or sessions between $\mathbf{V}^*$ and different honest incarnations of $\mathbf{P}$, $\mathbf{P}$ will generate (almost) independent random tapes to emulate the action of P_{R} in the second phase of the wrs-brZK argument in different sessions, and V_{R}^*, incorporating with P_{R}^{i,j_k}'s, uses the same strategy as $\mathbf{P}$ in the second phase of this argument. So, if the case 1 in item 5 does not occur, the only difference between $\mathbf{V}^*$'s view during the interaction with V_{R}^* and its view during the real interaction with many incarnations of $\mathbf{P}$ is that in the former interaction V_{R}^* uses pseudorandom generator to produce

YES instances y and uses the corresponding witness to execute the InstD rs-rWI argument. We can use a standard hybrid algorithm[3] to show $\mathbf{V}^*$'s view in these two scenario are indistinguishable, furthermore, note that V_{R}^*'s view is just the copy of $\mathbf{V}^*$'s view, so $(P_{\mathrm{R}}(\overline{w}), V_{\mathrm{R}}^*)(\overline{x})$ is computationally indistinguishable from $(\mathbf{P}(\overline{w}), \mathbf{V}^*)(\overline{x})$.

Note that the case 1 in item 5 occurs with only negligible probability when y is NO instance due to the uniqueness property of the InstD-VRF on NO key_instance. Therefore, if the case 1 occurs with non-negligible property in our setting that all y's are YES instances, then we construct an algorithm to break the pseudorandomness of the generator. This algorithm takes all witness for these statements $\overline{x}$ as inputs and uses them to execute the underlying InstD rs-rWI argument, if case 1 occurs, outputs "yes". It is not hard to see that this algorithm works.

Public-Coin Concurrent ZK Implies Simultaneously Resettable Secure ZK. If public-coin concurrent ZK argument exists, then we can construct key_instance-dependent resettably-sound *unbounded* resettable ZK argument and resettable WI argument with instance-dependent *unbounded* resettable soundness, therefore, using the above transformation, we get a resettably-sound resettable ZK argument. So, we have

Theorem 5. If there exist public-coin concurrent zero knowledge argument for a NP language L, and if trapdoor permutations exist, there exist resettably-sound resettable ZK argument for L.

We don't know if the public-coin concurrent ZK argument for non-trivial language exists, and if Barak's technique can be extended to (unbounded) concurrent setting. Theorem 5 shows this question deserves our attention: if such protocols (regardless of the number of rounds) for NP exist, the simultaneous resettability conjecture is true.

Acknowledgements Yi Deng is grateful to Boaz Barak and Yehuda Lindell for helpful conversations on concurrent and resettable zero knowledge. We thank anonymous reviewers for their encouragement and valuable comments.

References

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

[3] Consider an algorithm H takes the witnesses for all statement $\overline{x}$ as input, generates YES instances using a pseudorandom generator, and interacts with $\mathbf{V}^*$ using the witnesses for $\overline{x}$ as witnesses to execute the underlying InstD rs-rWI argument. Observe that $\mathbf{V}^*$'s view in the interaction with H is distinguishable form both its view in the interaction with V_{R}^* (due to resettable WI property of the underlying InstD rs-rWI argument) and its view in the interaction with $\mathbf{P}$ (due to pseudorandomness of the generator), thus we conclude the latter two are indistinguishable.

[2] B. Barak, O. Goldreich, S. Goldwasser, Y. Lindell. Resettably sound Zero Knowledge and its Applications. In Proc. of IEEE FOCS 2001, pp. 116-125.
[3] B. Barak, O. Goldreich. Universal Arguments and Their Applications. In Proc. of IEEE CCC 2002, pp. 194-203.
[4] M. Blum. How to Prove a Theorem so No One Else can Claim It. In Proc. of ICM'86, pp. 1444-1451, 1986.
[5] B. Barak, Y. Lindell, S. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. In Proc. of IEEE FOCS 2003, pp.384-393
[6] M. Blum, S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo Random Bits. In Proc. of IEEE FOCS 1982, pp. 112-117
[7] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In Proc. of IEEE FOCS 2001, pp.136-145
[8] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali. Resettable Zero Knowledge. In Proc. of ACM STOC 2000, pp.235-244
[9] 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.
[10] C. Dwork, M. Naor. Zaps and Their Applications. In Proc. of IEEE FOCS 2000, pp.283-293
[11] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In Proc. of ACM STOC 1998, pp.409-418.
[12] U.Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In Proc. of ACM STOC 1990, pp.416-426.
[13] O. Goldreich. Foundation of Cryptography-Basic Tools. Cambridge University Press, 2001.
[14] O. Goldreich, S. Goldwasser, S. Micali. How to construct random functions. J. ACM 33(4), pp.792-807
[15] 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.
[16] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM. J. Computing, 18(1):186-208, February 1989.
[17] J. Groth, R. Ostrovsky and A. Sahai. Non-interactive Zaps and New Techniques for NIZK. In Advances in Cryptology-Crypto'o6, LNCS 4117, pp.97-111.
[18] J. Hastad, R. Impagliazzo, L. A. Levin, M. Luby. A Pseudorandom Generator from Any One-Way Functions. SIAM Journal on Computing 28(4):1364-1396, 1999.
[19] T. Itoh, Y. Ohta. A language-dependent cryptographic primitive. Journal of Cryptology 10(1) pp.37-49, 1997
[20] Daniele Micciancio, Shien Jin Ong, Amit Sahai, Salil P. Vadhan. Concurrent Zero Knowledge Without Complexity Assumptions. TCC 2006, LNCS3876, pp.1-20
[21] S. Micali, L. Reyzin. Soundness in the public-key model. In Advances in Cryptology-Crypto'o2, LNCS2139, pp.542C565, 2001.
[22] S. Micali, R. Rivest. Micropayments revisited. In CT-RSA, pp.149C163, 2002.
[23] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In Proc. of IEEE FOCS, pp. 120C130, 1999.
[24] M. Naor. Bit Commitment using Pseudorandomness. Journal of Cryptology 4(2): 151-158, 1991.
[25] A. Yao. Theory and Applications of Trapdoor Functions. In Proc. of IEEE FOCS 1982, pp.80-91