Resettable Zero Knowledge in the Bare Public-Key Model under Standard Assumption

Yi Deng, Dongdai Lin

The state key laboratory of information security, Institute of software,

Chinese Academy of sciences, Beijing, 100080, China

Email: {ydeng,ddlin}@is.iscas.ac.cn

Abstract

In this paper we resolve an open problem regarding resettable zero knowledge in the bare public-key (BPK for short) model: Does there exist constant round resettable zero knowledge argument with concurrent soundness for $\mathcal{NP}$ in BPK model without assuming *sub-exponential hardness*? We give a positive answer to this question by presenting such a protocol for any language in $\mathcal{NP}$ in the bare public-key model assuming only collision-resistant hash functions against *polynomial-time* adversaries.

Key Words. Resettable Zero Knowledge, Concurrent Soundness, Bare Public-Key Model, Resettably sound Zero Knowledge.

1 Introduction

Zero knowledge (ZK for short) proof, a proof that reveals nothing but the validity of the assertion, is put forward in the seminal paper of Goldwasser, Micali and Rackoff [15]. Since its introduction, especially after the generality demonstrated in [14], ZK proofs have become a fundamental tools in design of some cryptographic protocols. To study the effect of executing ZK proofs in some realistic and asynchronous networks like the Internetin which Many instances of the same zero knowledge protocol may be executed concurrently, Dwork et al. [12]introduced the concept of concurrent zero knowledge. Though the concurrent zero knowledge protocols have wide applications, unfortunately, they requires

logarithmic rounds for languages outside $\mathcal{BPP}$ in the plain model for the black-box case [5] and therefore are of round inefficiency. In the Common Reference String model, Damgaard [6] showed that 3-round concurrent zero-knowledge can be achieved efficiently. Surprisingly, using non-black-box technique, Barak [1] constructed a constant round non-black-box bounded concurrent zero knowledge protocol which however is very inefficient.

Motivated by the application in which the prover (such as the user of a smart card) may encounter resetting attack, Canetti et al. [4] introduced the notion of resettable zero knowledge (rZK for short). An 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. Obviously the notion of resettable zero knowledge is stronger than that of concurrent zero knowledge and therefore we can not construct a constant round black-box rZK protocol in the plain model for non-trivial languages. To get constant round rZK, the work [4] also introduced a very attracting model, the bare public-key model(BPK). In this model, Each verifier deposits a public key pk in a public file and stores the associated secret key sk before any interaction with the prover begins. Note that no protocol needs to be run to publish sk, and no authority needs to check any property of pk. Consequently the BPK model is considered as a very weak set-up assumption compared to previously models such as common reference model and IPK model.

However, as Micali and Reyzin [18] pointed out, the notion of soundness in this model is more subtle. There are four distinct notions of soundness: one time, sequential, concurrent and resettable soundness, each of which implies the previous one. Moreover they also pointed out that there is NO black-box rZK satisfying resettable soundness for non-trivial language and the original rZK arguments in the BPK model of [4] does not seem to be concurrently sound. The first 4-round(optimal) rZK argument with concurrent soundness in the bare public-key model was proposed by Di Crescenzo et al. in [10].

All above rZK arguments need some cryptographic primitives secure against sub-exponential time adversaries,which is not a general assumption in cryptography. Using non-black-box techniques, Barak et al. obtained a constant-round rZK argument of knowledge assuming only collision-free hash functions secure against supperpolynomial-time algorithms[1], but their protocol enjoys only sequential soundness.

Our results. In this paper we present the first constant-round rZK argument with

[1]using idea from[3], this results also holds under standard assumptions that there exist hash functions that are collision-resistent against all polynomial-time adversaries.

concurrent soundness in BPK model for $\mathcal{NP}$ under the standard assumptions that there exist hash functions collision-resistant against *polynomial time* adversaries. We note that our protocol is a argument of knowledge and therefore the non-black-box technique is inherently used, and indeed we use the resettably-sound non-black-box zero knowledge argument [2] as a building block in which the verifier proves that a challenge is the one he committed to in a previous step. The key observation that enables the analysis of concurrent soundness is that it is not necessary in the proof of concurrent soundness to simulate all the concurrent execution of the underlying resettably-sound zero knowledge argument: we just need to simulate *only one execution* among all concurrent executions of the resettably-sound zero knowledge argument.

2 Preliminaries

In this section we recall some definitions and tools that will be used later.

In the following we say that function $f(n)$ is 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 denote by $\delta \leftarrow_R \Delta$ the process of picking a random element δ from Δ.

The BPK Model.The bare public-key model(BPK model)assumes that:

- A public file F that is a collection of records, each containing a verifier's public key, is available to the prover.
- 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 auxiliary input y, a public file F and a random tape r.
- An (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in two stages. In stage one, on input a security parameter 1^n and a random tape w, V generates a key pair (pk, sk) and stores pk in the file F. In stage two, on input sk, an n-bit string x and an random string w, V performs the interactive protocol with a prover, and outputs "accept x" or "reject x".

Definition 2.1 *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 y such that $(x, y) \in R_L$, here R_L is the relation induced by L, the probability that V interacting with P on input y, outputs "reject x" is negligible in n.*

Malicious 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.

P^* is a *s-concurrent malicious* prover if on input a public key pk of V, performs at most s interactive protocols as following: 1) if P^* is already running $i-1$ interactive protocols $1 \leq i-1 \leq s$, it can output a special message "Starting x_i," to start a new protocol with V on the new statement x_i; 2) At any point it can output a message for any of its interactive protocols, then immediately receives the verifier's response and continues.

A concurrent attack of a *s-concurrent malicious* prover P^* is executed in this way: 1) V runs on input 1^n and a random string and then obtains the key pair (pk, sk); 2) P^* runs on input 1^n and pk. Whenever P^* starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and sk.

Definition 2.2 $< P, V >$ *satisfies* concurrent soundness *for a language L if for all positive polynomials s, for all* s-concurrent malicious *prover P^*, the probability that in an execution of concurrent attack, V ever outputs "accept x" for $x \notin L$ is negligible in n.*

The notion of resettable zero-knowledge was first introduced in [4]. The notion gives a verifier the ability to rewind the prover to a previous state (after rewinding the prover uses the same random bits), and the *malicious* verifier can generate an arbitrary file F with several entries, each of them contains a public key generated by the malicious verifier. We refer readers to that paper for intuition of the notion. Here we just give the definition.

Definition 2.3 *An interactive argument system $< P, V >$ in the BPK model is black-box resettable zero-knowledge if there exists a probabilistic polynomial-time algorithm S such that for any probabilistic polynomial-time algorithm V^*, for any polynomials s, t, for any $x_i \in L$, the length of x_i is n, $i = 1, ..., s(n)$, V^* runs in at most t steps and the following two distributions are indistinguishable:*

1. *the view of V^* that generates F with $s(n)$ entries and interacts (even concurrently) a polynomial number of times with each $P(x_i, y_i, j, r_k, F)$ where y_i is a witness for $x_i \in L$, r_k is a random tape and j is the identity of the session being executed at present for $1 \leq i, j, k \leq s(n)$;*

2. *the output of S interacting with on input $x_1, ...x_{s(n)}$.*

Σ-protocols A protocol $< P, V >$ is said to be Σ-protocol for a relation R if it is of 3-move form and satisfies following conditions:

1. *Completeness*: for all $(x, y) \in R$, if P has the witness y and follows the protocol, the verifier always accepts.

2. *Special soundness*: Let (a, e, z) be the three messages exchanged by prover P and verifier V. From any statement x and any pair of accepting transcripts (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute y such that $(x, y) \in R$.

3. *Special honest-verifier ZK*: There exists a polynomial simulator M, which on input x and a random e outputs an accepting transcript of form (a, e, z) with the same probability distribution as a transcript between the honest P, V on input x.

Many known efficient protocols, such as those in [16] and [**?**], are Σ-protocols. Furthermore, there is a Σ-protocol for the language of Hamiltonian Graphs [1], assuming that one-way permutation families exists; if the commitment scheme used by the protocol in [1] is implemented using the scheme in [19] from any pseudo-random generator family, then the assumption can be reduced to the existence of one-way function families, at the cost of adding one preliminary message from the verifier. Note that adding one message does not have any influence on the property of Σ-protocols: assuming the new protocol is of form (f, a, e, z), given the challenge e, it is easy to indistinguishably generate the real transcript of form (f, a, e, z); given two accepting transcripts (f, a, e, z) and (f, a, e', z'), where $e \neq e'$, we can extract a witness easily. We can claim that any language in $\mathcal{NP}$ admits a 4-round Σ-protocol under the existence of any one-way function family (or under an appropriate number-theoretic assumption), or a Σ-protocol under the existence of any one-way permutation family. Though the following OR-proof refers only to 3-round Σ-protocol, readers should keep in mind that the way to construct the OR-proof is also applied to 4-round Σ-protocol.

Interestingly, Σ-protocols can be composed to proving the OR of atomic statements, as shown in [8, 7]. Specifically, given two protocols Σ_0,Σ_1 for two relationships R_0, R_1, respectively, we can construct a Σ_{OR}-protocol for the following relationship efficiently: $R_{OR} = ((x_0, x_1), y) : (x_0, y) \in R_0 or (x_1, y) \in R_1$, as follows. Let $(x_b, y) \in R_b$ and y is the private input of P. P computes a_b according the protocol Σ_b using (x_b, y). P chooses e_{1-b} and feeds the simulator M guaranteed by Σ_{1-b} with e_{1-b}, x_{1-b}, runs it and gets the output $(a_{1-b}, e_{1-b}, z_{1-b})$. P

sends a_b, a_{1-b} to V in first step. In second step, V picks $e \leftarrow_R \mathbb{Z}_q$ and sends it to P. Last, P sets $e_b = e \oplus e_{1-b}$, and computes the last message z_b to the challenge e_b using x_b, y as witness according the protocol Σ_b. P sends e_b, e_{1-b}, z_b) and e_{1-b}, z_{1-b} to V. V checks $e = e_b \oplus e_{1-b}$, and the two transcripts (a_b, e_b, z_b) and $(a_{1-b}, e_{1-b}, z_{1-b})$ are accepting. The resulting protocol turns out to be witness indistinguishable: the verifier can not tell which witness the prover used from a transcript of a session.

In our rZK argument, the verifier uses a 3-round Witness Indistinguishable Proof of Knowledge to prove knowledge of one of the two secret keys associating with his public key. As required in [11], we need a *partial-witness-independence* property from above proof of knowledge: the message sent at its first round should have distribution independent from any witness for the statement to be proved. We can obtain such a protocol using [22] [8].

Commitment scheme. A commitment scheme is a two-phase (committing phase and opening phase) two-party (a sender S and a receiver R)protocol which has following properties: 1) hiding: two commitments (here we view a commitment as a variable indexed by the value that the sender committed to) are computationally distinguishable for every probabilistic polynomial-time (possibly malicious) R^*; 2) Binding: after sent the commitment to a value m, any probabilistic polynomial-time (possibly malicious) sender S^* cannot open this commitment to another value $m' \neq m$ except with negligible probability. Under the assumption of existence of any one-way function families (using the scheme from [19] and the result from [17]) or under number-theoretic assumptions (e.g., the scheme from [**?**]), we can construct a schemes in which the first phase consists of 2 messages. Assuming the existence of one-way permutation families, a well-known non-interactive (in committing phase) construction of a commitment scheme (see, e.g. [13]) can be given.

A statistically-binding commitment scheme (with computational hiding) is a commitment scheme except with a stronger requirement on binding property: for all powerful sender S^* (without running time restriction), it cannot open a valid commitment to two different values except with exponentially small probability. We refer readers to [13] for the details for constructing statistically-binding commitments.

A perfect-hiding commitment scheme (with computational binding) is the one except with a stronger requirement on hiding property: the distribution of the commitments is indistinguishable for all powerful receiver R^*. As far as we know, all perfect-hinding commitment scheme requires interaction in the committing

phase.

Definition 2.4 *[13]. Let $d, r : N \to N$. we say that*

$$\{f_s : \{0,1\}^{d(|s|)} \to \{0,1\}^{r(|s|)}\}_{s \in \{0,1\}^*}$$

is an pseudorandom function ensemble if the following two conditions hold:

1. *1. Efficient evaluation: There exists a polynomial-time algorithm that on input s and $x \in 0, 1^{d(|s|)}$ returns $f_s(x)$;*

2. *2. Pseudorandomness: for every probabilistic polynomial-time oracle machine M, every polynomial $p(\cdot)$, and all sufficient large $n's$,*

$$|[Pr[M^{F_n}(1^n) = 1] - Pr[M^{H_n}(1^n) = 1]| < 1/p(n)$$

where F_n is a random variable uniformly distributed over the multi-set $\{f_s\}_{s \in \{0,1\}^n}$, and H_n is uniformly distributed among all functions mapping $d(n)$-bit-long strings to $r(n)$-bit-long strings.

3 A Simple Observation on Resettably-sound Zero Knowledge Arguments

resettably-sound zero knowledge argument is a zero knowledge argument with stronger soundness: for all probabilistic polynomial-time prover P^*, even P^* is allowed to reset the verifier V to previous state (after resetting the verifier V uses the same random tape), the probability that P^* make V accept a false statement $x \notin L$ is negligible.

In [2] Barak et al. transform a constant round public-coin zero knowledge argument $< P, V >$ for a $\mathcal{NP}$ language L into a resettably-sound zero knowledge argument $< P, W >$ for L [2]as follows: equip W with a collection of pseudorandom functions, and then let W emulate V except that it generate the current round message by applying a pseudorandom function to the transcript so far.

[2]In fact, Barak show how to transform a constant round public-coin argument for a relation R_L associated with the language L into a resettably-sound zero knowledge argument for R_L. However, we do not require the resettably-sound zero knowledge argument is a argument of knowledge when it is used as a building block in our main construction.

We will use a resettably-sound zero knowledge argument as a building block in which the verifier proves to the prover that a challenge is the one that he have committed to in previous stage. To prove our protocol showed in next section enjoys concurrent soundness, we will use the simulator associated with the resettably-sound zero knowledge argument to prove a false statement. The key observation that enables the analysis of concurrent soundness is that it is not necessary in the proof of concurrent soundness to simulate all the concurrent execution of the underlying resettably-sound zero knowledge argument: we just need to simulate *only one execution* among all concurrent executions of the resettably-sound zero knowledge argument. We call this property *one-many simulatability*. We note that Pass and Rosen [21] made a similar observation (in a different context) that enables the analysis of concurrent non-malleability of their commitment scheme.

Now we recall the Barak's constant round public-coin zero knowledge argument [1], and show this protocol satisfies *one-many simulatability*, and then so does the resettably-sound zero knowledge argument transformed from it.

Informally, Barak's protocol for a $\mathcal{NP}$ language L consists of two subprotocol: a general protocol and a WI universal argument. An real execution of the general protocol generates an instance that is unlikely in some properly defined language, and in the WI universal argument the prover proves that the statement $x \in L$ or the instance generated above is in the properly defined language. Let n be security parameter and $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a collection of hash functions where a hash function $h \in \mathcal{H}_n$ maps $\{0,1\}^*$ to $\{0,1\}^n$, and let C be a statistically binding commitment scheme. We define a language Λ as follows. We say a triplet $(h, c, r) \in \mathcal{H}_n \times \{o,1\}^n \times \{o,1\}^n$ is in Λ, if there exist a program Π and a string $s \in \{0,1\}^{poly(n)}$ such that $z = \mathsf{C}(h(\Pi), s)$ and $\Pi(z) = r$ within superpolynomial time (i.e., $n^{\omega(1)}$).

The Barak's Protocol [1])
Common input: an instance $x \in L$ ($|x| = n$)
Prover's private input: the witness w such that $(x, w) \in R_L$
$V \to P$: Send $h \leftarrow_R \mathcal{H}_n$;
$P \to V$: Pick $s \leftarrow_R \{0,1\}^{poly(n)}$ and Send $c = \mathsf{C}(h(0^{3n}, s)$;
$V \to P$: Send $r \leftarrow_R \{0,1\}^n$;
$P \Leftrightarrow V$: A WI universal argument in which P proves $x \in L$ or $(h, c, r) \in \Lambda$.

Fact 1. The Barak's protocol enjoys *one-many simulatability*. That is, For every malicious probabilistic polynomial time algorithm V^* that interacts with (arbitrary) polynomial s copies of P on true statements $\{x_i\}, 1 \leq i \leq s$, and for every

$j \in \{1, 2, ..., s\}$, there exists a probabilistic polynomial time algorithm S, takes V^* and all witness but the one for x_j, such that the output of $\mathsf{S}(V^*, \{(x_i, w_i)\}_{1 \leq i \leq s, i \neq j}, x_j)$ (where $(x_i, w_i) \in R_L$) and the view of V^* are indistinguishable.

We can construct a simulator $\mathsf{S} = (\mathsf{S}_{real}, \mathsf{S}_j)$ as follows: S_{real}, taking as inputs $\{(x_i, w_i)\}_{1 \leq i \leq s}$, does exactly what the honest provers do on these statements and outputs the transcript of all but the jth sessions (in jth session $x_j \in L$ is to be proven), S_j acts the same as the simulator associated with Barak's protocol in the session in which $x_j \in L$ is to be proven, except that when S_j is required to send a commitment value (the second round message in Barak's protocol), it commit to the hash value of the joint residual code of V^* and S_{real} at this point instead of committing to the hash value of the residual code of V^*, at the end of jth session S_j outputs the jth transcript. We note that the output of S_{real} is identical to the real interaction. For S_j, we can use the same analysis of the simulator associated with Barak's protocol to prove the output of S_j is indistinguishable from the real view of V^* in the jth session. Therefore, the simulator S satisfies our requirement. We sometimes call S a one-many simulator.

When we transform a constant round public-coin zero knowledge argument into a resettably-sound zero knowledge argument, the transformation itself does not influence the simulatability (zero knowledge) of the latter argument because the zero knowledge requirement does not refer to the honest verifier (as pointed out in [2]). Thus, the same simulator described above can be used to simulate one execution among all concurrent executions of the resettably-sound zero knowledge argument. So we have

Fact 2. All resettably-sound zero knowledge argument transformed from Barak's protocol enjoy *one-many simulatability*.

4 rZK Argument with Concurrent Soundness for $\mathcal{NP}$ in the BPK model Under Standard Assumption

In this section we present a constant-round rZK argument with concurrent soundness in the BPK model for all $\mathcal{NP}$ language without assuming any subexponential hardness.

For the sake of readability, we give some intuition before describe the protocol formally.

We construct the argument in the following way: build a concurrent zero knowledge argument with concurrent soundness and then transform this argument to a resettable zero knowledge argument with concurrent soundness. Concurrent zero knowledge with concurrent was first presented in [11] under standard assumption (without using "complexity leveraging"). For the sake of simplification, we modify the *flawed* construction presented in [23] to get a *correct* concurrent zero knowledge argument with concurrent soundness. Considering the following two-phase argument in BPK model: Let n be the security parameter, and f be a one way function that maps $\{0,1\}^{\kappa(n)}$ to $\{0,1\}^n$ for some function $\kappa : \mathbb{N} \rightarrow \mathbb{N}$. The verifier chooses two random numbers $x_0, x_1 \in \{0,1\}^{\kappa(n)}$, computes $y_0 = f(x_0)$, $y_1 = f(x_1)$ then publishes y_0, y_1 as he public key and keep x_0 or x_1 secret. In phase one of the argument, the verifier proves to the prover that he knows one of x_0, x_1 using a *partial-witness-independently* Witness Indistinguishable Proof of Knowledge. In phase two, the prover give a witness indistinguishable argument of knowledge that the statement to be proven is true or he knows one of x_0, x_1. Note that in phase one we use proof of knowledge while in phase two we use argument of knowledge, this means in phase two we restrict the prover to be a probabilistic polynomial-time algorithm, and therefore our whole protocol is an argument (not a proof).

Unfortunately, as pointed out in [11], the above two-phase argument does NOT enjoy concurrent soundness. In fact, when the *malicious* prover interacts with a verifier concurrently, he may decide a deliberate schedule so that he can learn how to prove that he knows one of x_0, x_1 from the proof given by the verifier in phase one. However, we can use the same technique in [11] in spirit to fix the flaw: in phase two, the prover uses a commitment scheme[3] COM_1 to compute a commitments to a random strings s, $c = \mathsf{COM}_1(s, r)$ (r is a random string needed in the commitment scheme), and then the prover prove that the statement to be proven is true or he committed to one of x_0, x_1 (i.e s equals x_0, or s equals x_1). We can prove that the modified argument is concurrent zero knowledge argument with concurrent soundness using technique similar to that in [11].

Given the above (modified) concurrent zero knowledge argument with concurrent soundness, we can transform it to resettable zero knowledge argument with concurrent soundness in this way: 1) using a statistically-binding commitment scheme COM_0, the verifier computes a commitment $c_e = \mathsf{COM}_0(e, r_e)$ (r_e is a

[3] In contrast to [11], we proved that computational binding commitment scheme suffices to achieve concurrent soundness. In fact, the statistically binding commitment scheme in [11] could also be replaced with computational binding one without violating the concurrent soundness.

random string needed in the scheme) to a random string e in the phase one, and then he sends e (note that the verifier does not send r_e, namely, it does not open the commitment c_e) as the second message (i.e the challenge) for the witness indistinguishable argument of knowledge used in the phase two and prove that e is the string he committed to in the first phase using resettably sound zero knowledge argument; 2)equipping the prover with a pseudorandom function, whenever the random bits is needed in a execution, the prover applied the pseudorandom function to what he have seen so far to generate random bits.

To prove the resulting argument enjoys concurrent soundness, we need to send a *false* challenge in phase 2 and simulate the malicious prover's view. The difficulty of the simulation lies in that the malicious prover interacts with the verifier in a interleaving way, and we do not know how to construct a constant round resettably sound argument with concurrent zero knowledge so far. However, we note that it is not necessary for the proof of concurrent soundness to simulate all the concurrent execution of the underlying resettably-sound zero knowledge argument, instead, we just need to simulate *only one execution* (i.e., the session that the malicious prover cheats the verifier) among all concurrent executions of the resettably-sound zero knowledge argument.

The Protocol (rZK argument with concurrent soundness in BPK model)

Let $\{prf_r : \{0,1\}^* \rightarrow \{0,1\}^{d(n)}\}_{r\in\{0,1\}^n}$ be a pseudorandom function ensembles, where d is a polynomial function, COM_0 be a *statistically-binding* commitment scheme, and let COM_1 be a general commitment scheme (can be either statistically-binding or computational-binding). Without loss of generality, we assume both the preimage size of the one-way function f and the message size of COM_1 equal n.

Common input: the public file F, n-bit string $x \in L$, an index i that specifies the i-th entry $pk_i = (f, y_0, y_1)$ (f is a one-way function) of F.

P**'s Private input:** a witness w for $x \in L$, and a fixed random string $(r_1, r_2) \in \{0,1\}^{2n}$.

V**'s Private input:** a secret key α (α=x_0 or x_1).

Phase 1:V Proves Knowledge of α and Sends a Committed Challenge to P.

1. V and P runs the 3-round *partial-witness-independently* witness indistinguishable protocol (Σ_{OR}-protocol) Π_v in which V prove knowledge of α that is one of the two preimages of y_0 and y_1. the randomness bits used by P equals r_1;

2. V computes $c_e = \mathsf{COM}_0(e, r_e)$ for a random e (r_e is a random string needed in the scheme), and sends c_e to P.

Phase 2: P Proves $x \in L$.

1. P checks the transcript of Π_v is accepting. if so, go to the following step.

2. P chooses a random string $s, |s| = n$, and compute $c = \mathsf{COM}_1(s, r_s)$ by picking a randomness r_s; P forms a new relation R'=$\{(x, y_0, y_1, c, w') \mid (x, w') \in R_L \vee (w' = (w'', r_{w''}) \wedge y_0 = f(w'') \wedge c = \mathsf{COM}_1(w'', r_{w''})) \vee (w' = (w'', r_{w''}) \wedge y_1 = f(w'') \wedge c = \mathsf{COM}_1(w'', r_{w''})))\}$; P invokes the 3-round witness indistinguishable argument of knowledge (Σ_{OR}-protocol) Π_p in which P prove knowledge of w' such that $(x, y_0, y_1, c; w') \in R'$, computes and sends the first message a of Π_p.
   All randomness bits used in this step is obtained by applying the pseudorandom function prf_{r_2} to what P have seen so far, including the common inputs, the private inputs and all messages sent by both parties so far.

3. V sends e to P, and execute a resettably sound zero knowledge argument with P in which V proves to P that $\exists\ r_e$ s.t. $c_e = \mathsf{COM}_0(e, r_e)$. Note that the subprotocol will costs several (constant) rounds. Again, the randomness used by P is generated by applying the pseudorandom function prf_{r_2} to what P have seen so far.

4. P checks the transcript of resettably sound zero knowledge argument is accepting. if so, P computes the last message z of Π_p and sends it to V.

5. V accepts if only if (a, e, z) is accepting transcript of Π_p.

Theorem 1. Let L be a language in $\mathcal{NP}$, If there exists hash functions collision-resistant against any polynomial time adversary, then there exists a constant round rZK argument with concurrent soundness for L in BPK model.

Remark on complexity assumption. We prove this theorem by showing the protocol described above is a rZK argument with concurrent soundness. Indeed, our protocol requires collision-resistant hash functions and one-way *permutations*, this is because the 3-round Σ-protocol (therefore Σ_{OR}-protocol) for $\mathcal{NP}$ assumes one-way permutations and the resettably sound zero knowledge

argument assumes collision-resistant hash functions. However, we can build 4-round Σ-protocol (therefore Σ_{OR}-protocol) for $\mathcal{NP}$ assuming existence of one-way functions by adding one message (see also discussions on Σ-protocol in section 2), and our security analysis can be also applied to this variant. We also note that collision-resistant hash functions implies one-way functions which suffices to build statistically-binding commitment scheme (therefore computational-binding scheme), thus, if we proved our protocol is a rZK argument with concurrent soundness, then we get theorem 1. Here we adopt the 3-round Σ_{OR}-protocol just for the sake of simplicity.

Proof. **Completeness.** Straightforward.

Resettable (*black-box*) resettable Zero Knowledge. The analysis is very similar to the analysis presented in [4, 10]. Here we omit the tedious proof and just provide some intuition. As usual, we can construct a simulator Sim that extracts all secret keys corresponding to those public keys registered by the malicious verifier from Π_v and then uses them as witness in executions of Π_p, and Sim can complete the simulation in expected polynomial time. We first note that when a malicious verifier resets a an honest prover, it can not send two different challenge for a fixed commitment sent in Phase 1 to the latter because of statistically-binding property of COM_0 and resettable soundness of the underlying sub-protocol used by the verifier to prove the challenge matches the value it has committed to in Phase 1. To prove the property of rZK, we need to show that the output of Sim is indistinguishable form the real interactions. However, in the execution of Π_p, the statement proved (i.e. (x, y_0, y_1, c)) by Sim and the one proved by a honest prover may be different (note that it is very unlikely for the simulator and the honest prover to generate the same commitment c), so the claim that the output of Sim is indistinguishable form the real interactions does not follow from the witness indistinguishability of Π_p directly. For this problem, We can construct a non-uniform hybrid simulator to overcome this difficulty: the non-uniform hybrid simulator taking as inputs all these secret keys and all the witnesses of statements in interactions, which computes commitments exactly as Sim does but executes Π_p using the same witness of the statement used by the honest prover. It is easy to see that the output of the hybrid simulator is indistinguishable from both the transcripts of real interactions (because of the computational-hiding property of COM_1) and the output of Sim (because of the witness indistinguishability of Π_p), therefore, we proved the the output of Sim is indistinguishable form the real interactions.

Concurrent Soundness. Proof proceeds by contradiction. The techniques

used here is similar to but different from that in [11].

Assume that the protocol does not satisfy the concurrent soundness property, thus there is a s-concurrently malicious prover P^*, concurrently interacting with V, makes the verifier accept a false statement $x_j \notin L$ in jth session with non-negligible probability p.

We now construct an algorithm B that takes the code (with randomness hard-wired in)of P^* and breaks the one-wayness of f with non-negligible probability.

B runs as follows. On input the challenge f, y (i.e given description of one-way function, B finds x such that $y = f(x)$), B randomly chooses $\alpha \in \{0,1\}^n$, $b \in \{0,1\}$, and guess a session number $j \in \{1, ..., s\}$(guess a session in which P^* will cheat the verifier successfully on a false statement x_j. Note that the event that this guess is correct happens with probability $1/s$), then B registers $pk = (f, y_0, y_1)$ as the public key, where $y_b = f(\alpha)$, $y_{1-b} = y$.

We write $\mathsf{B} = (\mathsf{B}_{real}, \mathsf{B}_j)$ without loss of generality. B interacts with P^* as honest verifier (note that B knows the secret key $sk = \alpha$ corresponding the public key pk) for all but jth session. Specifically, B employs the following rewinding strategy:

1. B acts as the honest verifier in the first time. That is, it completes Π_v using α as secret key, and commits to e, $c_e = \mathsf{COM}_0(e, r_e)$ in phase 1 then runs resettably sound ZK argument in Phase 2 using e, r_e as the witness. In particular, B uses B_j to play the role of verifier in the jth session, and uses B_{real} to play the role of verifier in all other sessions. At the end of jth session, B get an accepting transcript (a, e, z) of Π_p;

2. B_j rewind P^* to the point of beginning of step 3 in Phase 2 in jth session, it chooses a random string $e' \neq e$ and simulates the underlying resettably sound ZK argument in the same way showed in section 3: it commits to the hash value of the joint residual code of P^* and B_{real} in the second round of the resettably sound ZK argument (note this subprotocol is transformed from Barak's protocol) and uses them as the witness to complete the proof for the following *false* statement: $\exists\ r_e$ s.t. $c_e = \mathsf{COM}_0(e', r_e)$. If this rewinding incurs some other rewinds on other sessions, B_{real} always acts as an honest verifier, that is, it does not change the challenges that he committed to in Phase 1. When B get another accepting transcript (a, e', z') of Π_p at step 5 in Phase 2, it halts and outputs the two accepting transcript (a, e, z) and (a, e', z'), otherwise, B plays step 3 in jth session again.

Though the rewinding requires B_j simulates the P^*'s view on a *false* statement

(in section 3, the one-many simulation requires all statement is true)in jth session, however the simulation can be executes successfully and is indistinguishable from real interaction, and we can prove this using the same technique showed in the analysis of resettable zero knowledge property. We also note that if the simulation is successful, B gets another accepting transcript of Π_p with probability negligibly close to p. Since p is non-negligible, B will obtain two accepting transcript of Π_p with different challenges in expected polynomial time.

Now assume we extract an witness w' from the two different accepting transcripts of Π_p such that $(x, y_0, y_1, c, w') \in R'$, furthermore, the witness w' must satisfy $w' = (w'', r_{w''})$ and $y_b = f(w'')$ or $y_{1-b} = f(w'')$ because $x_j \notin L$. If $y_{1-b} = f(w'')$, we breaks the one-way assumption of f, otherwise(i.e., w'' satisfies $y_b = f(w'')$, in this case, $w'' = \alpha = x_b$, that is the secret key we knows), we fails. Next we claim B succeed in breaking the one-way assumption of f with non-negligible probability.

Assume otherwise, except with a negligible probability, B fails. This means if B always uses the witness x_b to execute Π_v during the above extraction, B always obtains two accepting transcript of Π_p in jth session in which it extracts x_b. Let k be the number of sessions executed before the end of session j. It is clear that B using k times the preimage of y_b (i.e., α) we always extract α, and B using k times the preimage of y_{1-b} we always extract the preimage of y_{1-b}. Thus, by hybrid argument, there must be $l \in \{1, ..., k\}$ such that using the preimage of y_b for first $l-1$ times and the preimage of y_{1-b} for last $k-l$ times, B will extract the preimage that is the same one used in execution of Π_v in lth session. We consider two cases: the execution of Π_v in lth session has been completed before or after step 2 in Phase 2 in jth session.

In case that the execution of Π_v in lth session has been completed before step 2 in Phase 2 in the jth session, if B outputs the preimage that is the same one used in execution of Π_v in lth session with probability negligibly close to 1, we can use this session to break the property of witness indistinguishability of Π_v because during the extraction we do not rewinds the lth session. This is impossible.

In case the execution of Π_v in lth session completed after the step 2 in Phase 2 in the jth session and the rewinding on jth session results in rewinding on execution of Π_v in lth session (if this is not the case, we have showed it is impossible in the first case), if B outputs the preimage that is the same one used in execution of Π_v in lth session with probability negligibly close to 1, We can construct a non-uniform algorithm B' to break the computational binding of the commitment scheme COM_1.

The non-uniform algorithm B' takes as auxiliary input $(y_b, y_{1-b}, x_b, x_{1-b})$ (with

input both secret keys) and interacts with P^* under the public key (y_b, y_{1-b}). It performs the following extraction:

1. Simulation: acts exactly as the B. Without loss of generality, let B' uses x_0 as witness in the lth execution of Π_v (i.e. the execution of Π_v in lth session), and B' obtains a accepting transcript (a, e_0, z_0) of Π_p in the jth session.

2. Rewinding Game 0: B' rewinds to the point of beginning of step 2 in Phase 2 in jth session and replays this round by sending another random challenge $e' \neq e$ until he gets another accepting transcript a, e'_0, z'_0 of Π_p using the same rewinding strategy of B and it uses x_0 as witness in the lth execution of Π_v which nested in the jth session.

3. Rewinding Game 1: repeats Rewinding Game 1 twice and obtains two accepting transcript (a, e_1, z_1) and (a, e'_1, z'_1) of Π_p, but B' uses x_1 as witness in the lth execution of Π_v during this game. Note that we assume the lth execution of Π_v completed after the step 2 in Phase 2 in the jth session, so at step 2 in Phase 2 P^* has seen at most the message in lth session (the first message of the protocol Π_v in lth session), and note that the Π_v is *partial-witness-independent* (so we can decide to use which witness at the last (third) step of Π_v in lth session) Σ-protocol, so B' can choose different witness to complete the lth execution of Π_v after P^* sent the first message a (the message in step 2 of Phase 2) of Π_p in the jth session.

Note that first message a sent by P^* in the jth session contains a commitment c and this message a (therefore c) remains unchanged during these games. Clearly, with non-negligible probability, B' will output two valid witness $w'_0 = ({w_0}'', r_{{w_0}''})$ and $w'_1 = ({w_1}'', r_{{w_1}''})$ from the above two games such that the following holds: $y_0 = f({w_0}'')$ (i.e. ${w_0}'' = x_0$), $y_1 = f({w_1}'')$ (i.e. ${w_1}'' = x_1$), $c = \mathsf{COM}_1({w_0}'', r_{{w_0}''})$ and $c = \mathsf{COM}_1({w_1}'', r_{{w_1}''})$. This contradicts the computational-binding property of the scheme COM_1.

In sum, we proved that for every case, it is impossible for B to output the same preimage that is used in execution of Π_v in lth session with probability negligibly close to 1, then the assumption that B fails except with a negligible probability is false, and this implies B succeeds in breaking the one-wayness of f with non-negligible probability. In another words, if the one-way assumption on f holds, it is infeasible for P^* to cheat an honest verifier in concurrent settings with non-negligible probability. □

Acknowledgments. The first author thanks Giovanni Di Crescenzo and Ivan Visconti for many helpful discussions and classifications.

References

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

[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] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali. Resettable Zero Knowledge. In Proc. of ACM STOC 2000.

[5] 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.

[6] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In Advances in Cryptology-EUROCYPT 2000, Springer LNCS 1807, pp.174-187.

[7] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Advances in Cryptology-CRYPTO'94, Springer Verlag LNCS 839, pp.174-187, 1994

[8] A. De Santis, G. Di Crescenzo, G. Persiano, M. Yung. On Monotone Formaula Close of SZK. In Proc. of IEEE FOCS 1994.

[9] G. Di Crescenzo, R. Ostrovsky. On Concurrent Zero Knowledge with Preprocessing. In Advances in Cryptology-Crypto 1999, Spriger LNCS1666, pp. 485-502.

[10] 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

[11] G. Di Crescenzo, Ivan Visconti. Concurrent Zero Knowledge in the Public-Key Model. In Proc. of ICALP 2005, Springer LNCS3580, pp.816-827.

[12] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In Proc. of ACM STOC 1998, pp.409-418.

[13] O. Goldreich. Foundation of Cryptography-Basic Tools. Cambridge University Press, 2001.

[14] 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.

[15] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM. J. Computing, 18(1):186-208, February 1989.

[16] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memery. In Advance in Cryptology-EUROCRYPT'88, Springer LNCS 330, pp.123-128, 1988.

[17] 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.

[18] S. Micali, L. Reyzin. Soundness in the Public-Key Model. In Advances in Cryptology-Crypto'01, Springer LNCS2139, pp.542-565.

[19] M. Naor. Bit Commitment using Pseudorandomness. Journal of Cryptology 4(2): 151-158, 1991.

[20] M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung: Perfect Zero-Knowledge Arguments for NP Using Any One-Way Permutation. Journal of 11(2): 87-108 (1998)

[21] R. Pass, A. Rosen: Concurrent Non-Malleable Commitments. In Proc. of IEEE FOCS 2005, pp.563-572, 2005

[22] C. P. Schnorr. Efficient Signature Generation for Smart Cards. Journal of Cryptology, 4(3): 239-252, 1991.

[23] Y. Zhao. Concurrent/Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model and its Applications. Cryptology ePrint Archive, Report 2003/265.