Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model and Its Applications

Yunlei Zhao[‡]

Abstract

In this work, we investigate concurrent knowledge-extraction (CKE) and concurrent non-malleability (CNM) for concurrent (and stronger, resettable) ZK protocols in the bare public-key model.

We formulate, driven by concrete attacks, and achieve CKE for constant-round concurrent/resettable arguments in the BPK model under standard polynomial assumptions. We get both generic and practical implementations. Here, CKE is a new concurrent verifier security that is strictly stronger than concurrent soundness in public-key model.

We then investigate, driven by concrete attacks, and clarify the subtleties in formulating CNM in the public-key model.

Keywords: concurrent knowledge-extraction, concurrent non-malleability, concurrent/resettable zero-knowledge, bare public-key model

[‡]Software School, School of Information Technology and Engineering, Fudan University, Shanghai 200433, P. R. China. ylzhao@fudan.edu.cn

0 Preface

This document records the evolution of the work originally appeared in the archive on 1 January 2004. With the current developments, the current title actually does not well reflect the contents of this work. A title, like "Concurrent Knowledge Extraction and Concurrent Non-Malleability in the Public-Key Model", may be more appropriate. We choose to use the old title to be compatible with previous references.

This work originally appears in the archive on 1st January 2004, and has underwent several main updates. Sections 4-5 have already appeared on the version of 19 January 2004, Section-6 appears since 7 June 2004, Section 7-9 appears since 19 August 2006 [1]. Sections 10 and 11 appear since 26 September 2006. We have chosen to update the work in an accumulating way such that each update remains previous contents intact. This makes the work appearing ugly, but can show a clear historical update records. For doubts about the historical update records, please request the archive editors for revealing past update records.

We note that the early versions of this work turn out to be flawed. Actually, as we shall see, the issues of concurrent knowledge extraction and concurrent non-malleability in public-key model are far more complex and subtle than one first glance. But, we remark that the early versions indeed contain some nice ideas and has triggered a number of subsequent works. To our knowledge, the early versions are also the first public attempt for achieving concurrent verifier security and concurrent non-malleability for *interactive* protocols in the public-key model under standard assumptions.

We are full of gratitude to Di Crescenzo, Persiano and Visconti for many helpful discussions on early versions of this work. The variant of FSZK in the public-key model is originally proposed by Lindell in January 2004, and we are indebted to his valuable suggestions. We thank Yung for numerous valuable discussions, constructive suggestions, and morale supports. We are grateful to A. C. Yao, Frances Yao, Xiaoyun Wang and Binyu Zang for valuable discussions and the highly precious supports that are critical for my research career. We also thank Robert H. Deng, Xiaotie Deng, Yi Deng, C. H. Lee, Wenbo Mao, Hong Zhu, and my students for many discussions, helps and supports.

[1] The formal presentation of Section 9, combined with partial results the work of [70], is submitted to TCC07 in a joint work with Yung.

1 Introduction

The notion of zero-knowledge (ZK) was introduced in the seminal paper of Goldwasser, Micali and Rackoff [48] to illustrate situations where a prover reveals nothing other than the verity of a given statement to an even malicious verifier. Since their introduction, zero-knowledge proofs have proven to be very useful as a building block in the construction of cryptographic protocols, especially after Goldreich, Micali and Wigderson [47] have shown that all languages in $\mathcal{NP}$ admit zero-knowledge proofs. By now, zero-knowledge has played a central role in the field of cryptography and is the accepted methodology to define and prove security of various cryptographic tasks.

Coin-tossing is one of the first and more fundamental protocol problem in the literature [12]. In its simplest form, the task calls for two mutually distrustful parties to generate a common random string [9]. Recent research efforts have intensively investigated efficient coin-tossing in more complicated settings. For example, in the secure two-party computation setting, constant-round secure coin-tossing (and accordingly, constant-round secure two-party computation by combining the results of [68, 46]) is firstly achieved by Lindell [53]. In the "man-in-the-middle (MIM)" setting, constant-round non-malleable coin-tossing protocol in the plain model (and accordingly, constant-round non-malleable zero-knowledge arguments for $\mathcal{NP}$ and commitment schemes by combining the result of [24]) is firstly achieved by Barak [2].

With the emergence and far and wide sweeping popularity of the Internet, much recent research attention, initiated by Dwork, Naor and Sahai [35], has been paid to the security threats of cryptographic protocols when they are executing concurrently in an asynchronous network setting like the Internet. In this scenario, many concurrent executions of the same protocol take place in an asynchronous network setting. All communication channels are assumed unauthenticated and controled by an adversary. Honest players are assumed oblivious of each other's existence, nor do they generally know the topology of the network, and thus cannot coordinate their executions. However, a malicious adversary that interacts with a number of players can schedule all the executions concurrently at its wish. There are three security threats to be addressed for concurrent executions of a two-party cryptographic protocol in unauthenticated and asynchronous network settings: concurrent non-malleability (channel security threat), concurrent left-player security threat (e. g. concurrent zero-knowledge for a zero-knowledge protocol) and concurrent right-player security threat (e. g. concurrent soundness for a zero-knowledge protocol). We call a cryptographic protocol concurrently secure if it is immune against all the above three kinds of security threats. In particular, concurrently secure coin-tossing protocols are named *concurrent coin-tossing* in this work.

1.1 A historical view of related works

1.1.1 Related works on concurrent and resettable zero-knowledge

The study of security under concurrent composition is initiated by Dwork, Naor and Sahai in the context of concurrent zero-knowledge [35]. Although non-constant-round black-box concurrent zero-knowledge proofs for $\mathcal{NP}$ exist in the plain model under standard intractability assumptions [65, 50, 51, 64, 57], but they cannot be constant-round in the black-box sense [14, 15]. Constant-round non-black-box bounded concurrent zero-knowledge arguments and arguments of knowledge for $\mathcal{NP}$ are achieved in [1, 5]. To achieve constant-round black-box concurrent zero-knowledge, several computatinal models are introduced: the timing model [35, 45], the preprocessing model [32], the common reference string model[19], and the bare public-key model [13].

The bare public-key (BPK) model is first introduced by Canetti, Goldreich, Goldwasser and Micali [13] to achieve round-efficient resettable zero-knowledge (rZK) that is a generalization and strengthening of the notion of concurrent zero-knowledge. As pointed out by Micali and Reyzin [58], although

introduced with a specific application in mind, the BPK model applies to interactive systems in general, regardless of their knowledge complexity. A protocol in BPK model simply assumes that all verifiers have deposited a public key in a public file before any interaction takes place among the users. This public file is accessible to all users at all times. Note that an adversary may deposit many (possibly invalid or fake) public keys in it, particularly, without even knowing corresponding secret keys or whether such exist. That is, no trusted third party is assumed in the BPK model, the prover is not involved in the preprocessing, and there is also no assumption on the asynchronousity of the communication network. Consequently, the BPK model is considered a weaker setup assumption with respect to the models proposed in [35, 45, 32, 19].

The BPK model is thus very simple, and it is in fact a weak version of the frequently used public-key infrastructure (PKI) model, which underlies any public-key cryptosystem or digital signature scheme. Despite its apparent simplicity, the BPK model is quite powerful. While cZK and rZK protocols exist both in the standard and in the BPK models [13], only in the latter case they can be constant-round, at least in the black box sense. Various soundness notions of cryptographic protocols in public-key models are noted and clarified by Micali and Reyzin [58]. In public-key models, a verifier V has a secret key SK, corresponding to its public-key PK. A malicious prover P^* could potentially gain some knowledge about SK from an interaction with the verifier. This gained knowledge might help him to convince the verifier of a false theorem in another interaction. Micali and Reyzin showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger one-time, sequential, concurrent and resettable soundness. In this paper we focus on concurrent soundness which roughly means, for zero-knowledge protocols, that a malicious prover P^* can not convince the honest verifier V of a false statement enen P^* is allowed multiple interleaved interactions with V. Micali and Reyzin [58] showed that the constant-round rZK argument in the BPK model present in [13] and their improvement [58] enjoy sequential soundness while they conjecture that both protocols do not satisfy concurrent soundness, thus they are not secure in an asynchronous setting as the Internet. Three-round resettable zero-knowledge with concurrent soundness in some stronger version of the BPK model can be found in [59, 75].

For arguments of knowledge, rZK (non-black-box) arguments of knowledge for $\mathcal{NP}$ is achieved by Barak, Goldreich, Goldwasser and Lindell [4].

We remark that all the concurrent zero-knowledge protocols mentioned in this subsection are not provably concurrent non-malleability. That is, these works do not explicitly deal with the concurrent channel security .

1.1.2 Related works on non-malleability

Non-malleability in the "man-in-the-middle" setting is first studied by Dolve, Dwork and Naor in [33] and there they also give firstly the CCA-2 non-malleable public-key cryptosystem, non-constant-round (stand-alone) non-malleable zero-knowledge protocols for $\mathcal{NP}$ and non-constant-round (stand-alone) non-malleable commitment schemes in the plain model under standard complexity assumptions. Their results also imply a non-constant-round (stand-alone) non-malleable coin-tossing protocol in the plain model. Constant-round (stand-alone) non-malleable coin-tossing protocol in the plain model is first achieved by Barak by following the technique presented in [53] for constant-round secure coin-tossing and using non-black-box simulation [1].

Non-malleability in the common reference string model has been extensively investigated recently. Sahai introduced non-malleable NIZK in [66] where he shows how to construct NIZK which remains non-malleable only as long as the number of proofs seen by any adversary is bounded. Unbounded (concurrent) non-malleable NIZK for $\mathcal{NP}$ in the common reference string model is achieved by De Santis et al. [24]. .

Non-interactive non-malleable commitment schemes in the common reference string model is achieved by Di Crescenzo, Ishai, and Ostrovsky [30] assuming the existence of one-way functions. More efficient constructions based on specific assumptions can be found in [31, 41]. The constructions from [30, 31, 41] are proved secure according to a stand-alone non-malleability definition where the adversary sees *one* commitment from the honest player and then tries to make its own (maliciously related) commitment. Unbounded (concurrent) non-malleability of commitment schemes in the common reference string model is achieved very recently by Damgard and Groth [21] where they named this notion of security *reusability*.

1.2 Our contributions

In this paper, we define concurrent coin-tossing and show how to implement it in constant rounds in the bare public-key (BPK) model [13] under standard intractability assumptions. We stress that the registered public-keys in the BPK model are not used in any way to achieve an authentication scheme, even for concurrent non-malleability where players are assumed to be honest and so the public-keys are well-formed. In comparison with the work of Barak [2], the work of [2] is the first constant-round stand-alone non-malleable coin-tossing protocol in the plain model (without any trusted third party or setup assumption). The work of [2] assumes subexponential hardness assumptions and runs in at least 10 rounds since the work of [1] uses ZK universal arguments and the known ZK universal arguments runs in 10 rounds. emploies non-black-box and complexity leveraging techniques. Our work deals with concurrently secure coin-tossing (that implies concurrently non-malleable coin-tossing) in the BPK model. Our work does not assume any sub-exponential hardness assumption and runs in 8 rounds. and also does not employ non-black-box and complexity leveraging techniques in the security proofs.

A critical tool for achieving concurrent coin-tossing in the BPK model developed in this paper is (general and practical) constant-round CZK-CS protocols.

Our concurrent coin-tossing implies concurrently secure zero-knowledge protocols, specifically constant-round concurrent zero-knowledge argument of knowledge for $\mathcal{NP}$ with concurrent soundness and concurrent non-malleability, in the BPK model by composing our result with the robust NIZK in the common random string model [24]. Our result also implies constant-round concurrently non-malleable commitment schemes (non-malleable with respect to commitments) in the BPK model. There are two ways to achieve constant-round concurrently non-malleable commitment schemes in the BPK model. One way is to combine our result with the (very recently developed) reusable and non-malleable commitment schemes in the common random string model [21]. Another way is to first commit to a value and then use a constant-round concurrently non-malleable zero-knowledge argument of knowledge protocol to prove knowledge of the committed value.

To our knowledge, there is no previous concurrent non-malleability result known beyond the common reference string model. Note that in contrary to the common reference string model, where a trusted third party is implicitly assumed besides the honest players, *no* trusted third party is assumed in the BPK model. Note that non-malleability refers to the unauthenticated channel security among honest players.

1.3 Organization

In Section 2, we present preliminaries. In section 3, we present the definitions of concurrent zero-knowledge and concurrent soundness in the BPK model. In Section 4, we give constant-round general and practical CZK-CS protocols in the BPK model. In Section 5, we present some further improvements on the round-complexity and on rZK-CS. In Section 6, we define and construct general and practical coin-tossing protocol.

2 Preliminaries

In this section, we quickly recall the major cryptographic tools used.

We use standard notations and conventions below for writing probabilistic algorithms and experiments. If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots ; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots ; r)$. If S is a finite set then $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement.

2.1 Σ-protocols for proving the knowledge of commitment trapdoors

Σ-protocols are first introduced by Cramer, Damgard and Schoenmakers [18]. Informally, a Σ-protocol is itself a 3-round public-coin special honest verifier zero-knowledge protocol with special soundness in the knowledge-extraction sense. Since its introduction, Σ-protocols have been proved a very powerful cryptographic tool and are widely used in numerous important cryptographic applications including digital signatures (by using the famous Fiat-Shamir methodology [42] and efficient electronic payment systems [17]. For a good survey of Σ-protocols and their applications, readers are referred to [20, 17].

Definition 2.1 (Σ-protocol [18]) *A 3-round public-coin protocol $< P, V >$ is said to be a Σ-protocol for relation R if the following hold:*

- *Completeness. If P, V follow the protocol, the verifier always accepts.*
- *Special soundness. From any common input x and any pair of accepting conversations on input x, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently computes w such that $(x, w) \in R$. Here a, e, z stand for the first, the second and the third message respectively.*
- *Special honest verifier zero-knowledge (SHVZK). There exists a polynomial-time simulator S, which on input x and a random challenge string e, outputs an accepting conversation of the form (a, e, z), with the same probability distribution as conversations between the honest P, V on input x.*

Definition 2.2 (trapdoor commitment scheme TC) *A trapdoor commitment scheme (TC) is a quintuple of probabilistic polynomial-time (PPT) algorithms TCGen, TCCom, TCVer, TCKeyVer and TCFake, such that*

- *Completeness.* $\forall n$, $\forall v$, $\Pr[(TCPK, TCSK) \stackrel{R}{\leftarrow} \text{TCGen}(1^n); (c, d) \stackrel{R}{\leftarrow} \text{TCCom}(TCPK, v) : \text{TCKeyVer}(TCPK, 1^n) = \text{TCVer}(TCPK, c, v, d) = \text{YES}] = 1$.
- *Computational Binding. For all sufficiently large n and for all PPT adversaries A, the following probability is negligible in n:* $\Pr[(TCPK, TCSK) \stackrel{R}{\leftarrow} \text{TCGen}(1^n); (c, v_1, v_2, d_1, d_2) \stackrel{R}{\leftarrow} \text{A}(1^n, TCPK) :$

  $\text{TCVer}(TCPK, c, v_1, d_1) = \text{TCVer}(TCPK, c, v_2, d_2) = \text{YES}$ and $v_1 \neq v_2]$.
- *Perfect Hiding.* $\forall TCPK$ *such that* $\text{TCKeyVer}(TCPK, 1^n) = \text{YES}$ *and* $\forall v_1, v_2$ *of equal length, the following two probability distributions are identical:* $[(c_1, d_1) \stackrel{R}{\leftarrow} \text{TCCom}(TCPK, v_1) : c_1]$ *and* $[(c_2, d_2) \stackrel{R}{\leftarrow} \text{TCCom}(TCPK, v_2) : c_2]$.
- *trapdoorness.* $\forall (TCPK, TCSK) \in \{\text{TCGen}(1^n)\}$, $\forall v_1, v_2$ *of equal length, the following two probability distributions are identical:*

$[(c, d_1) \stackrel{R}{\leftarrow} \text{TCCom}(TCPK, v_1); d_2' \stackrel{R}{\leftarrow} \text{TCFake}(TCPK, TCSK, c, v_1, d_1, v_2) : (c, d_2')]$ *and*
$[(c, d_2) \stackrel{R}{\leftarrow} \text{TCCom}(TCPK, v_2) : (c, d_2)]$.

The following is a construction of trapdoor commitment scheme based on DLP intractability assumption [11]: On a security parameter n, the receiver selects uniformly an n-bit prime p so that $q = (p-1)/2$ is a prime, an element g of order q in $\mathbf{Z}_p^*$. Then the receiver uniformly selects w in $\mathbf{Z}_q^*$ and sets $h = g^w \ mod \ p$. The receiver publishes (p, q, g, h) as its public-key and keeps w as its secret-key (i. e. the trapdoor). To commit a bit σ, the sender first checks that (p, q, g, h) is of the right form (otherwise it halts announcing that the receiver is cheating), uniformly selects $s \in \mathbf{Z}_q$, and sends $g^s h^\sigma \ mod \ p$ as its commitment.

Feige-Shamir Trapdoor Commitments

The following one-way function based (computational hiding and computational binding) trapdoor commitment scheme is firstly introduced by Feige and Shamir [40], which is based on the zero-knowledge proof for DHC (directed Hamiltonicity cicle) of Blum [10].

Key Generation. Let f be a one-way function, then on a security parameter n, the commitment verifier randomly chooses $x \in \{0,1\}^n$ and computes $y = f(x)$. Then by using the (Cook-Levin) $\mathcal{NP}$-reduction the commitment verifier reduces the language $\{y | \exists x s.t y = f(x)\}$ to Hamiltonicity, to obtain a graph G (with $p(n)$ nodes) so that finding a Hamiltonian cycle in G is equivalent to finding the preimage x of y, where $p(n)$ is a positive polynomial in n. Note that the one-wayness of f implies the difficulty of finding a Hamiltonian cycle in G. The commitment verifier published the graph G, or equivalently the string y, as its public-key and keeps x in secret as its secret-key. Note that, from x it is easy to generate a Hamiltonian cycle in G.

Commitments and decommitments. To commit to 0, the commitment prover chooses a random permutation π, permutes the nodes of G, and commits to the entries of the resulting adjacency matrix by using the one-round OWF-based perfect-binding commitment scheme defined in ?. The commitment prover reveals the committed bit '0' by revealing π and the entries of the matrix.

To commit to 1, the commitment prover chooses the $p(n)$ node clique and commits to its adjacency matrix (which is all 1) by using the one-round OWF-based perfect-binding commitment scheme. The commitment prover reveals the committed bit '0' by openning a random cycle in this matrix.

Trapdoorness. Given a Hamiltonian cycle in G, it is possible to generates commitments that are indistinguishable from legal ones, and yet have the property that one can decommit to both 0 and 1. In particular, after committing to a random permutation of G, it is possible to decommit to 0 in the same ways. However, it is also possible to decommit to 1 by only revealing the (known) Hamiltonian cycle in G.

We remark that since the underlying OWF-based one-round perfect-binding commitment scheme is only computationally hiding, the above trapdoor commitment scheme is also computationaly hiding. In the rest of this paper, we denote by FSTC the above OWF-based (both computational binding and computational hiding) trapdoor commitment scheme.

And the following is a Σ-protocol $< P, V >$ suggested by Schnorr [67] for proving the knowledge of trapdoor secret-key, w, for a public-key of the above form (p, q, g, h) such that $h = g^w \ mod \ p$:

- P chooses r at random in $\mathbf{Z}_q$ and sends $a = g^r \ mod \ p$ to V.
- V chooses a challenge e at random in $\mathbf{Z}_{2^t}$ and sends it to P. Here, t is fixed such that $2^t < q$.

- P sends $z = r + ew \ mod \ p$ to V, who checks that $g^z = ah^e \ mod \ p$, that p, q are prime and that g, h have order q, and accepts iff this is the case.

The OR-proof of Σ-protocols. As shown in [18], any public-coin SHVZK protocol is itself witness indistinguishable (WI). Although for languages that each instance has a single witness, Σ-protocols for that languages is trivially WI, one basic construction with Σ-protocols allows a prover to show that given two inputs x_0, x_1 that each of them has a single witness, he knows w such that either $(x_0, w) \in R$ or $(x_1, w) \in R$, BUT without revealing which is the case.

So we assume we are given a Σ-protocol $< P, V >$ for R with random challenges of length t. Assume also that (x_0, x_1) are common input to P, V, and that w is private input to P, where $(x_b, w) \in R$ for $b = 0$ or 1. Roughly speaking, the idea is that we will ask the prover to complete two instances of $< P, V >$, with respect to x_0, x_1 respectively. For x_b, he can do this for real, for x_{1-b} he will have to fake it using the SHVZK simulator. However, if we give him a little freedom in choosing the challenges to answer, he will be able to complete both instances. More precisely, consider the following protocol, which we call Σ_{OR}:

- P computes the first message a_b in $< P, V >$, using x_b, w as private inputs. P chooses e_{1-b} at random and runs the SHVZK simulator S on input x_{1-b}, e_{1-b}, let $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output. P finally sends a_0, a_1 to V.
- V chooses a random t-bit string s and sends it to P.
- P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b to challenge e_b using (x_b, a_b, e_b, w) as input. He sends (e_0, z_0, e_1, z_1) to V.
- V checks that $s = e_0 \oplus e_1$ and that conversations (a_0, e_0, z_o), (a_1, e_1, z_1) are accepting conversations with respect to inputs x_0, x_1, respectively.

Theorem 2.1 [20] *The protocol Σ_{OR} above is a Σ-protocol for R_{OR}, where $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R \text{ or } (x_1, w) \in R\}$. Moreover, for any verifier V^*, the probability distribution of conversations between P and V^*, where w is such that $(x_b, w) \in R$, is independent of b. That is, Σ_{OR} is perfectly witness indistinguishable.*

2.2 Other cryptographic tools

We proceed to present other cryptographic tools used in this paper.

Definition 2.3 (system for proof of knowledge) *Let R be a binary relation and $\kappa : N \to [0, 1]$. We say that a probabilistic polynomial-time (PPT) interactive machine V is a **knowledge verifier for the relation** R with **knowledge error** κ if the following two conditions hold:*

- *Non-triviality: There exists an interactive machine P such that for every $(x, y) \in R$ all possible interactions of V with P on common input x and auxiliary input y are accepting.*
- *Validity (with error κ): There exists a polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every interactive machine P^*, every $x \in L_R$, and every $y, r \in \{0, 1\}^*$, machine K satisfies the following condition:*

  Denote by $p(x, y, r)$ the probability that the interactive machine V accepts, on input x, when interacting with the prover specified by $P^*_{x,y,r}$ (where $P^*_{x,y,r}$ denotes the strategy of P^* on common input x, auxiliary input y and random-tape r). If $p(x, y, r) > \kappa(|x|)$, then, on input x and with

oracle access to $P^*_{x,y,r}$, machine K outputs a solution $s \in R(x)$ within an accepted number of steps bounded by

$$\frac{q(|x|)}{p(x,y,r) - \kappa(|x|)}$$

*The oracle machine K is called a **knowledge extractor**.*

An interactive proof system (P,V) such that V is a knowledge verifier for a relation R and P is a machine satisfying the non-triviality condition (with respect to V and R) is called a system for proof of knowledge for the relation R. The proof system (P,V) is a system of zero-knowledge proof of knowledge (ZKPOK) if it is also zero-knowledge.

For more clarifications on the definition of proof of knowledge, readers are referred to [44]. More recent advances of zero-knowledge arguments of knowledge can be found in [53, 5].

Definition 2.4 (witness indistinguishability WI) *Let $< P, V >$ be an interactive proof system for a language $L \in \mathcal{NP}$, and let R_L be the fixed $\mathcal{NP}$ witness relation for L. That is $x \in L$ if there exists a w such that $(x, w) \in R_L$. We denote by $view^{P(w)}_{V^*(z)}(x)$ a random variable describing the transcript of all messages exchanged between V^* and P in an execution of the protocol on common input x, when P has auxiliary input w and V^* has auxiliary input z. We say that $< P, V >$ is witness indistinguishability for R_L if for every PPT interactive machine V^*, and every two sequences $W^1 = \{w^1_x\}_{x\in L}$ and $W^2 = \{w^2_x\}_{x\in L}$, so that $(x, w^1_x) \in R_L$ and $(x, w^2_x) \in R_L$, the following two probability distributions are computationally indistinguishable: $\{x, view^{P(w^1_x)}_{V^*(z)}\}_{x\in L,\, z\in\{0,1\}^*}$ and $\{x, view^{P(w^2_x)}_{V^*(z)}\}_{x\in L,\, z\in\{0,1\}^*}$.*

In this paper we use 3-round public-coin witness indistinguishability proofs of knowledge (WIPOK) for $\mathcal{NP}$. Such protocols exists under the existence of one-way functions, e. g. the parallel repetitions of Blum's 3-round proof of knowledge for HC [**?**]. We remark that 2-round public-coin WI proofs for $\mathcal{NP}$ do exist under the existence of one-way permutations [34].

Definition 2.5 (non-interactive zero-knowledge NIZK) *Let NIP and NIV be two interactive machines and NIV is also probabilistic polynomial-time, and let $NI\sigma Len$ be a positive polynomial. We say that $\langle NIP, NIV\rangle$ is an NIZK proof system for an $\mathcal{NP}$ language L, if the following conditions hold:*

- *Completeness. For any $x \in L$ of length n, any σ of length $NI\sigma Len(n)$, and $\mathcal{NP}$-witness w for x, it holds that*
$$\Pr[\Pi \xleftarrow{R} NIP(\sigma, x, w) : NIV(\sigma, x, \Pi) = YES] = 1.$$
- *Soundness. $\forall x \notin L$ of length n,*
$$\Pr[\sigma \xleftarrow{R} \{0,1\}^{NI\sigma Len(n)} : \exists\, \Pi \; s.t. \; NIV(\sigma, x, \Pi) = YES] \text{ is negligible in } n.$$
- *Zero-Knowledgeness. $\exists$ a PPT simulator NIS such that, $\forall$ sufficiently large n, $\forall x \in L$ of length n and $\mathcal{NP}$-witness w for x, the following two distributions are computationally indistinguishable:*
$$[(\sigma', \Pi') \xleftarrow{R} NIS(x) : (\sigma', \Pi')] \text{ and } [\sigma \xleftarrow{R} \{0,1\}^{NI\sigma Len(n)}; \Pi \xleftarrow{R} NIP(\sigma, x, w) : (\sigma, \Pi)].$$

Non-interactive zero-knowledge proof systems for $\mathcal{NP}$ can be constructed based on any one-way permutation [39]. An efficient implementation based on any one-way permutation is presented in [**?**] and readers are referred to [24] for recent advances of NIZK.

Definition 2.6 (NIZK proof of knowledge [?]) *An NIZK proof system $\langle NIP, NIV\rangle$ for a language $L \in \mathcal{NP}$ with witness relation R_L (as defined above) is NIZK proof of knowledge (NIZKPOK) if there exists a pair of PPT machines (E_1, E_2) and a negligible function ε such that for all sufficiently large n:*

- *Reference-String Uniformity. The distribution on reference strings produced by $E_1(1^n)$ has statistical distance at most $\varepsilon(n)$ from the uniform distribution on $\{0,1\}^{NI\sigma Len(n)}$.*
- *Witness Extractability. For all adversaries A, we have that $\Pr[\mathbf{Expt}_A^E(n) = 1] \geqslant \Pr[\mathbf{Expt}_A(n) = 1] - \varepsilon(n)$, where the experiments $\mathbf{Expt}_A(n)$ and $\mathbf{Expt}_A^E(n)$ are defined as follows:*

$\mathbf{Expt}_A(n)$:	$\mathbf{Expt}_A^E(n)$:
$\sigma \xleftarrow{R} \{0,1\}^{NI\sigma Len(n)}$	$(\sigma, \tau) \longleftarrow E_1(1^n)$
$(x, \Pi) \longleftarrow A(\sigma)$	$(x, \Pi) \longleftarrow A(\sigma)$
return $NIV(x, \sigma, \Pi)$	$w \longleftarrow E_2(\sigma, \tau, x, \Pi)$
	return 1 if $(x, w) \in R_L$

NIZK proofs of knowledge for $\mathcal{NP}$ can be constructed assuming the existence of one-way permutations and dense secure public-key cryptosystems [**?**].

3 Definitions of concurrent zero-knowledge and concurrent soundness in the BPK model

In this section, we present the formal definitions of concurrent zero-knowledge and concurrent soundness in the BPK model.

Concurrent zero-knowledge in the BPK model. Let $\bar{\mathbf{x}} = \{x_1, x_2, \cdots, x_q\}$, where $|x_1| = |x_2| = \cdots = |x_q|$ and q is a polynomial in n. Upon $\bar{\mathbf{x}}$, an adversary V^* in the BPK model firstly outputs an arbitrary public-file F that includes a list of (without loss of generality) q public-keys $pk_1, \cdots, pk_q$. Then V^* concurrently interacts with q^2 instances of the honest prover: $P(x_i, pk_j)$, $1 \leqslant i, j \leqslant q$, and schedules all the concurrent executions at its wish. We remark that each instance of the honest prover uses independent random strings. Without loss of generality we also assume that messages from V^* are immediately answered by the honest prover instances.

Definition 3.1 *We say that a proof or argument system $< P, V >$ for a language L in the BPK model is black-box concurrent zero-knowledge if there exists a probabilistic polynomial-time (PPT) oracle machine S (the simulator) such that for any polynomial q in n and for any PPT adversary V^*, the distributions $< P, V^* > (\bar{\boldsymbol{x}})$ and $S^{V^*}(\bar{\boldsymbol{x}})$ are computationally indistinguishable for any sequence of common inputs $\bar{\boldsymbol{x}} = x_1, x_2, \cdots, x_q \in L \cap \{0,1\}^n$.*

Concurrent soundness in the BPK model. For an honest verifier V with public-key PK and secret-key SK, an (s,t)-concurrent malicious prover P^* in the BPK model, for a pair positive polynomials (s,t), be a probabilistic $t(n)$-time Turing machine that, on a security parameter 1^n and PK, performs concurrently at most $s(n)$ interactive protocols (sessions) with V as follows.

If P^* is already running $i-1$ $(0 \leq i-1 < s(n))$ sessions, it can select *on the fly* a common input $x_i \in \{0,1\}^n$ (which may be equal to x_j for $1 \leq j < i$) and initiate a new session with $V(SK, x_i)$. We note that in different sessions V uses independent random-tapes.

We then say a protocol satisfies *concurrent soundness* in the BPK model if for any honest verifier V, for all positive polynomials (s,t), for all (s,t)-concurrent malicious prover P^*, the probability that there exists i $(1 \leq i \leq s(n))$ such that $V(SK, x_i)$ outputs "accept x_i" while $x_i \notin L$ is negligible in n.

4 Constructions for Constant-Round Concurrent Zero-Knowledge With Concurrent Soundness in the BPK model

In this section, we present general and practical constant-round concurrent zero-knowledge argument of knowledge with concurrent soundness for $\mathcal{NP}$ in the BPK model.

4.1 The general construction

The general construction $< P, V >$ is depicted in Figure 1 (page 11).

The protocol $< P, V >$

Key Generation. For a security parameter n, let $(TCPK_0, TCSK_0) \stackrel{\mathrm{R}}{\leftarrow} \mathrm{TCGen}(1^n, r_0)$, $(TCPK_1, TCSK_1) \stackrel{\mathrm{R}}{\leftarrow} \mathrm{TCGen}(1^n, r_1)$, where r_0 and r_1 are two independent random strings used by TCGen. $(TCPK_0, TCPK_1)$ is the public-key of the verifier V. But for its secret-key, the verifier V randomly selects a bit $b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}$ and keeps $TCSK_b$ in secret as its secret-key while discards $TCSK_{1-b}$.
The public file F in the BPK model is a collection of records (id, PK_{id}), where $PK_{id} = (TCPK_0^{(id)}, TCPK_1^{(id)})$ is the alleged public-key of the verifier with identity id, V_{id}. The secret-key of V_{id}, SK_{id}, is $TCSK_b^{(id)}$ for a random bit b in $\{0,1\}$.

Common input. An element $x \in L \cap \{0,1\}^n$. Denote by R_L the corresponding $\mathcal{NP}$-relation for L.

P private input. An $\mathcal{NP}$-witness y for $x \in L$.

Stage 1. The verifier V proves the knowledge that: he knows either $TCSK_0$ or $TCSK_1$ with respect to his public-key $(TCPK_0, TCPK_1)$, by using the Σ_{OR} protocol of Schnorr's Σ-protocol (described in Section 2.1) for proving commitment trapdoors. The witness used by V is its secret-key $TCSK_b$.

Stage 2. The prover P uniformly selects two independent random strings $r_P^{(0)} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{NI\sigma Len(n)}$ and $r_P^{(1)} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{NI\sigma Len(n)}$, and for two independent random strings $s^{(0)}$, $s^{(1)}$, computes $\alpha_0 = \mathrm{TCCom}(TCPK_0, r_P^{(0)}, s^{(0)})$ and $\alpha_1 = \mathrm{TCCom}(TCPK_1, r_P^{(1)}, s^{(1)})$ using the trapdoor commitment scheme TC. Finally, the left player sends (α_0, α_1) to the right player.

Stage 3. The verifier V uniformly selects $r_V \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{NI\sigma Len(n)}$ and sends r_V to P.

Stage 4. P sends $r = r_P^{(i)} \oplus r_V$ to the right player for $i \stackrel{R}{\longleftarrow} \{0,1\}$.

Stage 5. Using a 3-round public-coin WIPOK for $\mathcal{NP}$, P proves that either α_0 or α_1 commits to $r \oplus r_V$. That is, P proves the knowledge of $(i, r \oplus r_V, s)$ such that $i \in \{0,1\}$ and $\alpha_i = \mathrm{TCCom}(TCPK_i, r \oplus r_V, s)$. The witness used by P is $(i, r_P^{(i)}, s^{(i)})$ for.

Stage 6. Using r as the common input, P gives a NIZKPOK that he knows y such that $(x, y) \in R_L$. [a]

[a] We remark it is not necessarily to use NIZK proof of knowledge in Stage 6. Actually, a NIZK argument of knowledge for $\mathcal{NP}$ does also work here. For example, we can adopt the robust NIZK (presented in [24]) that is a same-string unbounded non-malleable NIZK argument of knowledge for $\mathcal{NP}$. Here same-string NIZK means that the reference string generated by the simulator of robust NIZK is a uniformly random string rather than a pseudorandom one as usual. The same-string property can be used to simplify future security analyses.

Figure 1. A constant-round concurrent zero-knowledge argument of knowledge with concurrent soundness for $\mathcal{NP}$ in the BPK model

We remark that the above general protocol is similar to the concurrent zero-knowledge protocol in the timing model developed by Dwork and Naor [34]. The double commitments technique can be traced to [60] in achieving public-key cryptography secure against chosen message attacks. This technique is also used in other works (e. g. [25, 66, 24, 54]).

The protocol depicted in Figure 1 runs in 8 rounds. But it can be reduced into 6 rounds by accordingly combining some rounds. Specifically, the Stage 2 and Stage 3 can be combined into the Stage 1.

Theorem 4.1 *Under Discrete Logarithm assumption and the existence of trapdoor one-way permutations and dense secure public-key cryptosystems, the protocol depicted in Figure 1 is a constant-round concurrent zero-knowledge argument of knowledge with concurrent soundness for $\mathcal{NP}$ in the BPK model.*

Proof. The completeness of the protocol can be easily checked. Below, we focus on the properties of concurrent zero-knowledge and concurrent soundness.

Black-box concurrent zero-knowledge

For any adversary V^* described in Section 3.1, we need to construct a PPT simulator S such that the output of S^{V^*} is computationally indistinguishable from the view of V^* in its real concurrent interactions with honest-prover instances.

The simulation procedure is similar to (but simpler than) the simulation procedure presented in [13] for resettable zero-knowledge. Specifically, S works in at most $q+1$ rounds, where q is the number of public-keys registered by V^* in the public-key file. In each round, S either successfully gets a simulated transcript or "breaks" a new public-key in the sense that S can extract the corresponding secret-key $TCSK_b$ (according to the special soundness of Σ-protocol, this is achieved by rewinding V^* to get two accepting conversations of Stage 1). Once a public-key is broken (that is S learns $TCSK_b$), then in any session with respected to this broken public-key S works as follows: S runs accordingly just as a honest prover in Stage 1-3; Let (α_0, α_1) be the message sent by S in Stage 2 (that commit to two independent random values, $r_P^{(0)}$ and $r_P^{(1)}$, respectively) and r_{V^*} be the message sent by V^* in Stage 3; In Stage 4, S runs the NIZK simulator to get a random string, denoted σ, and sends σ to V^*; In Stage 5, S uses $(b, \sigma \oplus r_{V^*}, s')$ as its witness to give a WIPOK, where $s' \stackrel{\mathrm{R}}{\leftarrow} \mathrm{TCFake}(TCPK_b, TCSK_b, \alpha_b, r_P^{(b)}, s^{(b)}, \sigma \oplus r_{V^*})$. Finally, S using the NIZK simulator to give a simulated NIZK (on σ) in Stage 6.

Since S runs in at most q rounds, at during each round S also works in expected polynomial time, it is easy to see that S also runs in expected polynomial time in toto. Below, we show that the output of S is indistinguishable from the view of V^* in real interactions.

We consider four classes of transcripts: they differ according to the value sent in Stage 4 ($r_P^{(i)} \oplus r_{V^*}$ or σ generated by NIZK simulator), the witness used in Stage 5, the Stage-6 message (real NIZK or simulated NIZK).

1. Stage-4 message is $r_P^{(i)} \oplus r_{V^*}$ for $i \stackrel{R}{\longleftarrow} \{0,1\}$, Stage-5 witness is $(\alpha_i, r^{(i)}, s^{(i)})$. Stage-6 message is a real NIZK.

2. Stage-4 message is $r_P \oplus r_{V^*}$, Stage-5 witness is $(\alpha_b, r_P, s_{r_P}^{(b)})$, where b is the secret information of V^* in its secret-key $TCSK_b$. Stage-6 message is a real NIZK.

3. Stage-4 message is σ, Stage-5 witness is $(\alpha_b, \sigma \oplus r_{V^*}, s')$. Stage-6 message is a real NIZK.

4. Stage-4 message is σ, Stage-5 witness is $(\alpha_b, \sigma \oplus r_{V^*}, s')$. Stage-6 message is a simulated NIZK.

The real transcripts are the first class. The simulator outputs the fourth class. It is easy to see that Class 3 and Class 4 are computationally indistinguishable. Class 1 and Class 2 are computationally indistinguishable by the witness indistinguishability of Stage 5. We note that Class 3 and Class do not make (non-negligibly) noticeable distinguishability gap. Actually, if we adopt robust NIZK in Stage 6

as suggested in the protocol construction, the distributions of Class 2 and Class 3 are identical. The reason is that the σ generated by the simulator of robust NIZK is uniformly distributed. This means σ, $\sigma \oplus r_{V^*}$, r_P and $r_P \oplus r_{V^*}$ are all uniformly random strings. Furthermore, according to the trapdoor property of the trapdoor commitment scheme used, $s_{r_P}^{(b)}$ and s' are also independent random strings.

Concurrent soundness and argument of knowledge

We first note that a computational power unbounded prover can easily convince the verifier of a false statement since he can get the secret-keys if his computational power is unbounded. Hence the protocol $< P, V >$ depicted in Figure 1 constitutes an argument system rather than a proof system.

We now proceed to prove that the protocol $< P, V >$ is concurrent soundness and argument of knowledge. The following proof uses standard reduction and knowledge-extraction techniques . That is, if the protocol $< P, V >$ does not satisfy concurrent soundness in the BPK model then we will construct a non-uniform algorithm S that breaks the discrete logarithm assumption in expected polynomial-time.

Suppose the protocol $< P, V >$ does not satisfy concurrent soundness in the BPK model, then according to the definition of concurrent soundness in the BPK model (described in Section 3), then in a concurrent attack issued by an (s,t)-concurrent malicious prover P^* against an honest verifier V with public-key $(TCPK_0, TCSK_1)$ and secret-key $TCSK_b$ for $b \xleftarrow{R} \{0,1\}$, with non-negligible probability $p(n)$ there exists an i, $1 \leqslant i \leqslant s(n)$, such that V outputs "accept x_i" while $x_i \notin L$. Then we will construct a non-uniform algorithm S that takes $(TCPK_0, TCPK_1, TCSK_b)$ as input and outputs either a witness w such that $(x_i, w) \in R_L$ or $TCSK_{1-b}$ with probability $\frac{p^4(n)}{4s(n)}$ in polynomial-time. For this purpose, S has oracle access to P^* and plays the role of V by running $V(TCPK_0, TCPK_1, TCSK_b)$ to emulate the real concurrent interactions between P^* and $V(TCPK_0, TCPK_1, TCSK_b)$.

For any i, $1 \leqslant i \leqslant s(n)$, denote by $k_i \in \{0,1\}$ the first component in the witness used by P^* in Stage-5 of i-th session. We first have the following lemma:

Proposition 4.1 *For any i, $1 \leqslant i \leqslant s(n)$, k_i is independent of b. That is, $\Pr[k_i = b] = \frac{1}{2}$, where the probability is over the coin flips of P^* and V.*

Proof. We can view the honest V in two parts: the first part, denoted V_1, only works in Stage-1. That is, V_1 takes $TCSK_b$ as its secret input and proves that he knows one of the secret-key with respect to the public-key $(TCPK_0, TCPK_1)$ in Stage-1. Note that V_1 is itself perfectly witness indistinguishable; the second part, denote V_2, works in other stages and has no secret information.

Now, suppose there exists an i, $1 \leqslant i \leqslant s(n)$, such that k_i is not independent of b, then we can construct a PPT algorithm A that works as follows to violate the perfect witness indistinguishability of V_1: A interacts with V_1 on common input $(TCPK_0, TCPK_1)$ and runs P^* and V_2 to emulate the concurrent interactions between P^* and V. After the simulation, A randomly guess the "bad" i and outputs k_i as its output. Suppose P^* distinguishes b with probability p then A will distinguish b with probability $p/s(n)$, which violates the perfect WI property of V_1. □

For future reference convenience, below we denote by Stage 5.1, 5.2 and 5.3 the first, the second and the third message of Stage 5 in protocol $< P, V >$ respectively, where Stage 5.2 is supposed to be a random string. Let (E_1, E_2) be the pair of PPT algorithms guaranteed in the definition of NIZKPOK. Now, consider the following algorithm $S^{P^*}(1^n, TCPK_0, TCPK_1, TCSK_b)$ depicted in Figure 2 (page 24).

First, we note that P^* cannot distinguish (except with statistically negligible probability) whether he is interacting with honest verifier V or with S since the distribution of σ generated by E_1 is statistically distinguishable from the uniform distribution on $\{0,1\}^{NI\sigma Len(n)}$. This means that , in the concurrent

The algorithm $S^{P^*}(1^n, TCPK_0, TCPK_1, TCSK_b)$

$i \xleftarrow{R} \{1, 2, \cdots, s(n)\}$.

Runs P^* and acts accordingly by running $V(TCPK_0, TCPK_1, TCSK_b)$ in any session other than the i-th session. In the i-th session, denote by $(\alpha_0^{(i)}, \alpha_1^{(i)})$ the Stage-2 message of the i-th session, S acts as follows:

- Uniformly selects $r_V \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{NI\sigma Len(n)}$ and sends r_V to P^* as the Stage-3 message of the i-th session.
- When running into the WIPOK phase (Stage 4-5) of the i-th session, denoted by r the Stage-4 message (from P^*) of the i-th session, S uses the knowledge-extractor of WIPOK to extract the witness used by P^* in Stage-5 of the i-th session. This needs to rewind P^* once and such a rewinding is called the first *knowledge rewinding*. If the extracted value is (k_i, t_i, s_i) such that $\alpha_{k_i}^{(i)} = \mathrm{TCCom}(TCPK_{k_i}, t_i, s_i)$ and $t_i = r \oplus r_V$, where t_i is of length $NI\sigma Len(n)$ and $k_i \in \{0,1\}$, then S does the following:
  1. runs E_1 to get $(\sigma, \tau) \longleftarrow E_1(1^n)$.
  2. rewinds P^* to the point P^* just sent $(\alpha_0^{(i)}, \alpha_1^{(i)})$ and sends back $\sigma \oplus t_i$ to P^* as a new Stage-3 message. Such a rewinding is called the *major rewinding*.
  3. Runs P^* further (from the major rewinding point). When running into again the WIPOK phase (Stage 4-5), *if the Stage-4 message from P^* is not σ* then S uses the knowledge-extractor of WIPOK again to extract the witness used by P^* in this WIPOK phase. This needs to knowledge-rewind P^* once more. Denote by (k_i', t_i', s_i') the witness extracted in the second knowledge-rewinding.
  4. If P^* successfully gives an NIZKPOK Π on σ for x_i at Stage-6, S runs $E_2(\sigma, \tau, x_i, \Pi)$ to get a witness w.

Figure 2. The algorithm $S^{P^*}(1^n, TCPK_0, TCPK_1, TCSK_b)$

interactions between the (s,t)-concurrent malicious P^* and S, with the same non-negligible probability $p(n)$ there exists an i, $1 \leqslant i \leqslant s(n)$, such that S outputs "accept x_i" while $x_i \notin L$. Furthermore, conditioned on P^* always succeeds in completing its interactions with S, then $Pr[k_i = k_i' = 1 - b] = \frac{1}{4}$ according to the proposition 4.1. Since i is uniformly selected by S from $\{1, 2, \cdots, s(n)\}$, we conclude that with probability at least $\frac{p^4}{4s(n)}$, S either get a witness w such that $(x_i, w) \in R_L$ or two different decommitments to $\alpha_{1-b}^{(i)}$ from which the secret-key $TCSK_{1-b}$ can be easily extracted. This contradicts to the assumption that $x_i \notin L$ and discrete logarithm is hard. Thus the protocol $< P, L >$ is concurrently sound in the BPK model.

That the system is an argument of knowledge is immediately from the extraction procedure of S. □

4.2 The practical construction

We remark that by using the techniques presented in [57], the above 6-round general protocol can be transformed into a practical protocol without going through the general $\mathcal{NP}$-reductions. Specifically, for any language that admits a Σ-protocol, we present for the same language a 6-round black-box concurrent zero-knowledge argument of knowledge with concurrent soundness in the BPK model. Let $\langle P_L, V_L \rangle$ be a Σ-protocol for a language L and denote by (p_1, q, p_2) be the messages exchanged between honest P_L and honest V_L on a common input $x \in L \cap \{0,1\}^n$, where q is suggested to be an n-bit value randomly chosen according to some distribution. Denote by S_L be the honest verifier zero-knowledge simulator of $\langle P_L, V_L \rangle$. We transform $\langle P_L, V_L \rangle$ into a protocol $\langle P, V \rangle$ in the BPK model that is depicted in Figure 3 (page 17). The protocol $\langle P, V \rangle$ uses the DLP-based trapdoor commitment scheme TC and we denote by S_{OR} the honest verifier zero-knowledge simulator of the Σ_{OR} protocol of Schnorr's protocol for discrete logarithm.

The practical protocol $\langle P, V \rangle$

Key Generation. For a security parameter n, let $(TCPK_0, TCSK_0) \stackrel{\mathrm{R}}{\leftarrow} \mathrm{TCGen}(1^n, r_0)$, $(TCPK_1, TCSK_1) \stackrel{\mathrm{R}}{\leftarrow} \mathrm{TCGen}(1^n, r_1)$, where r_0 and r_1 are two independent random strings used by TCGen. $(TCPK_0, TCPK_1)$ is the public-key of the verifier V. But for its secret-key, the verifier V randomly selects a bit $b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}$ and keeps $TCSK_b$ in secret as its secret-key while discards $TCSK_{1-b}$.
The public file F in the BPK model is a collection of records (id, PK_{id}), where $PK_{id} = (TCPK_0^{(id)}, TCPK_1^{(id)})$ is the alleged public-key of the verifier with identity id, V_{id}. The secret-key of V_{id}, SK_{id}, is $TCSK_b^{(id)}$ for a random bit b in $\{0,1\}$.

Common input. An element $x \in L \cap \{0,1\}^n$. Denote by R_L the corresponding $\mathcal{NP}$-relation for L.

P private input. An $\mathcal{NP}$-witness y for $x \in L$.

Stage 1. The verifier V proves the knowledge that: he knows either $TCSK_0$ or $TCSK_1$ with respect to his public-key $(TCPK_0, TCPK_1)$, by using the Σ_{OR} protocol of Schnorr's Σ-protocol (described in Section 3.1) for proving discrete logarithm. The witness used by V is its secret-key $TCSK_b$.

Stage 2. The prover P uniformly selects two independent random strings $r_P^{(0)} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and $r_P^{(1)} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$, and for two independent random strings $s^{(0)}$, $s^{(1)}$, computes $\alpha_0 = \mathrm{TCCom}(TCPK_0, r_P^{(0)}, s^{(0)})$ and $\alpha_1 = \mathrm{TCCom}(TCPK_1, r_P^{(1)}, s^{(1)})$ using the trapdoor commitment scheme TC. Finally, the prover sends (α_0, α_1) to the verifier.

Stage 3. The verifier V uniformly selects $r_V \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and sends r_V to P.

Stage 4. Stage 4 includes the following three steps:

Stage 4.1. Using the simulator S_{OR} (of the Σ_{OR} protocol of Schnorr's Σ-protocol) on input $(\alpha_0, \alpha_1, r_V)$, the prover obtains a transcript $(\hat{a}, \hat{e}, \hat{z})$. (Informally, here the prover uses the simulator S_{OR} to "pretend" that one of (α_0, α_1) commits to r_V). Then P runs P_L to compute p_1 and sends $(\hat{a}, p_1)$ to V.

Stage 4.2. The verifier sends back P a random value q' of length n.

Stage 4.3. The prover computes $q = \hat{e} \oplus q'$ and $p_2 = P_L(x, y, p_1, q)$. P then sends $(\hat{e}, \hat{z}, p_2)$ to the verifier.

Verifier's decision The verifier accepts if and only if $(\hat{a}, \hat{e}, \hat{z})$ is an accepting conversation on $(\alpha_0, \alpha_1, r_V)$ and $(p_1, \hat{e} \oplus q', p_2)$ is an accepting conversation on input x.

Figure 3. The practical construction of zero-knowledge with concurrent player security in the BPK model for any language that admits Σ-protocols.

We remark that the protocol depicted in Figure 3 runs in 8 rounds. But it can be reduced into 6 rounds by accordingly combining some rounds. Specifically, the Stage-2 and Stage-3 can be combined into the Stage-1. We also remark that the protocol can be easily extended to transform any public-coin honest verifier zero-knowledge protocol into concurrent zero-knowledge in the BPK model with three

additional rounds. But for simplicity, we only present the transformation starting from any Σ-protocol which is the most often case when public-coin honest verifier zero-knowledge protocols are used in practice.

Theorem 4.2 *Under the discrete logarithm assumption and $\langle P_L, V_L\rangle$ is a Σ-protocol for L, the protocol depicted in Figure 3 is a 6-round black-box concurrent zero-knowledge argument of knowledge for L with concurrent soundness in the BPK model but without going through general $\mathcal{NP}$-reductions. Furthermore, if $\langle P_L, V_L\rangle$ has the property of honest verifier* perfect *zero-knowledge then the protocol is also concurrent* perfect *zero-knowledge.*

Proof (sketch).

(1) completeness.

If $x \in L$ then P can always complete the proof and V accepts it.

(2) Black-box concurrent zero-knowledge.

For any adversary V^* described in Section 4.1.3, we need to construct a PPT simulator S such that the output of S^{V^*} is computationally indistinguishable from the view of V^* in its real concurrent interactions with honest-prover instances.

The simulation procedure is similar to (but simpler than) the simulation procedure presented in [13] for resettable zero-knowledge. Specifically, S works in at most $s(n)+1$ phases, where $s(n)$ is the number of public-keys registered by V^* in the public-key file. In each phase, S either successfully gets a simulated transcript or "breaks" a new public-key in the sense that S can extract the corresponding secret-key $TCSK_b$ (according to the special soundness of Σ-protocol, this is achieved by rewinding V^* to get two different accepting conversations w. r. t. the same first message of Stage 1). Once a public-key $(TCPK_0, TCPK_1)$ is broken (that is, S learns $TCSK_b$), then in any session with respected to this broken public-key S works as follows:

S runs accordingly just as an honest prover in Stage 1-3. Let (α_0, α_1) be the message sent by S in Stage 2 (that commit to two independent random values, $r_P^{(0)}$ and $r_P^{(1)}$, respectively) and r_{V^*} be the message sent by V^* in Stage 3. In Stage 4, S firstly runs the honest verifier zero-knowledge simulator of the Σ-protocol $\langle P_L, V_L\rangle$ to get a simulated transcript (p_1, q, p_2). Then in Stage 4.1, S sends $(\hat{a}, p_1)$ to V^*. After receiving back a random value q' from V^* in Stage 4.2, S sets $\hat{e} = q \oplus q'$, computes $\hat{z}$ on $(\alpha_0, \alpha_1, r_{V^*}, \hat{a}, \hat{e})$ by using $TCSK_b$ as its witness, and finally sends $(\hat{e}, \hat{z}, p_2)$ to V^* in Stage 4.3.

Since S runs in at most $s(n)+1$ phases, and during each phase S also works in expected polynomial time, it is easy to see that S runs in expected polynomial time in toto. Below, we show that the output of S is indistinguishable from the view of V^* in real interactions.

We consider three classes of transcripts: they differ according to the (simulated or real) transcript $(\hat{a}, \hat{e}, \hat{z})$ of the Σ_{OR} protocol of Schnorr's Σ-protocol, and the (real or simulated) transcript (p_1, q, p_2) of the Σ-protocol $\langle P_L, V_L\rangle$ in Stage 4 of each session.

1. Simulated $(\hat{a}, \hat{e}, \hat{z}) = S_{OR}(\alpha_0, \alpha_1, r_{V^*}, \hat{e})$; real $(p_1, q = q' \oplus \hat{e}, p_2)$ (generated by using y as the witness).

2. Real $(\hat{a}, \hat{e}, \hat{z})$ (generated by using $TCSK_b$ as the witness); real $(p_1, q = q' \oplus \hat{e}, p_2)$ (generated by using y as the witness).

3. Real $(\hat{a}, \hat{e} = q \oplus q', \hat{z})$ (generated by using $TCSK_b$ as the witness) and simulated $(p_1, q, p_2) = S_L(x, q)$.

The real transcript of concurrent interactions between V^* and honest prover instances is the first class. The simulator outputs the third class. The second class is the transcript of the following mental experiment executed between V^* and honest provers: In each session of the mental experiment w. r. t. a public-key $(TCPK_0, TCPK_1)$ and a common input x, the honest prover P takes both the witness y such that $(x, y) \in R_L$ and the corresponding secret-key $TCSK_b$ as its auxiliary inputs. In stage 4.1, P sends $(\hat{a}, p_1)$ to V^*. After receiving q' from V^* in stage 4.2, P randomly selects $\hat{e}$, computes $\hat{z}$ on $(\alpha_0, \alpha_1, r_{V^*}, \hat{a}, \hat{e})$ by using $TCSK_b$ as its witness, sets $q = q' \oplus \hat{e}$ and computes p_2 on (x, p_1, q) by using y as the witness, and finally sends $(\hat{e}, \hat{z}, p_2)$ to V^* in stage 4.3.

We first observe that in the first class, according to the honest verifier perfect zero-knowledge property of the Σ_{OR} of Schnorr's protocol for proving discrete logarithms, q is also truly random. In other words, upon seeing the simulated value $\hat{a}$, V^* cannot in any way choose the q' dependently on $\hat{e}$. For the same reason, in the third class, $\hat{e}$ is also at least pseudorandom. Note that in the second class, both q and $\hat{e}$ are truly random. Then, by using standard hybrid techniques, it is easy to see that Class 1 and Class 2 are identical, and Class 3 and Class 2 are also indistinguishable. Furthermore, if $\langle P_L, V_L \rangle$ has the property of honest verifier *perfect* zero-knowledge, then Class 3 and Class 2 are also identical. This means that in this case the protocol presented in Figure 1 is black-box concurrent *perfect* zero-knowledge.

(3) Concurrent soundness and argument of knowledge.

We first note that a computational power unbounded prover can easily convince the verifier of a false statement since he can extract the secret-keys if his computational power is unbounded. Hence the protocol $\langle P, V \rangle$ depicted in Figure 3 constitutes an argument system rather than a proof system.

We now proceed to prove that the protocol $\langle P, V \rangle$ satisfies concurrent soundness in the BPK model. The following proof uses standard reduction and knowledge-extraction techniques. Specifically, suppose the protocol $\langle P, V \rangle$ does not satisfy concurrent soundness in the BPK model, then according to the definition of concurrent soundness in the BPK model (described in Section 4.1.2), in a concurrent attack issued by an (s, t)-concurrent malicious prover P^* against an honest verifier V with public-key $(TCPK_0, TCSK_1)$ and secret-key $TCSK_b$ for $b \xleftarrow{R} \{0, 1\}$, with non-negligible probability $p(n)$ there exists a k, $1 \leqslant k \leqslant s(n)$, such that V outputs "accept x_k" while $x_k \notin L$. Then we will construct an algorithm S that takes a public-key $TCPK$ as input and outputs either a witness for $x_k \in L$ or the corresponding $TCSK$ with probability at least $\frac{(p(n))^4}{4s(n)}$ in polynomial-time, which breaks either the assumption that $x_k \notin L$ or the discrete logarithm hardness assumption. The algorithm S is depicted in Figure 4 (page 20).

According to the description of S in Figure 4, conditioned on P^* always succeeds in completing its interactions with S, then both in the interactions prior to the major rewinding and in the interactions posterior to the major rewinding of the rewound i-th session there are two cases to be considered:

1. S gets two different accepting conversations of the Σ-protocol $\langle P_L, V_L \rangle$ on input x_i with respect to the same Stage 4.1 message (specifically, the same p_1 message). According to the special soundness property of Σ-protocol, this means a witness of $x_i \in L$ can be efficiently extracted.

2. S gets two different accepting conversations of the Σ-protocol Σ_{OR} on input $(\alpha_0, \alpha_1, r_V)$ with respect to the same Stage 4.1 message (specifically, the same $\hat{a}$ message). According to the special soundness property of Σ-protocol, this means that a witness of the form (s, j) can be efficiently extracted, where $j \in \{0, 1\}$ and $\alpha_j = TCCom(TCPK_j, r_V, s)$.

The algorithm $S^{P^*}(1^n, TCPK)$

$(TCPK', TCSK') \xleftarrow{R} TCGen(1^n)$.
$b \xleftarrow{R} \{0,1\}$.
Set $TCPK_b$ be $TCPK'$, $TCSK_b$ be $TCSK'$ and $TCPK_{1-b}$ be $TCPK$. S publishes $(TCPK_0, TCPK_1)$ as its public-key and keeps $TCSK_b = TCSK'$ in secret as its secret-key.

$i \xleftarrow{R} \{1, 2, \cdots, s(n)\}$.

S runs P^* and acts accordingly by running $V(TCPK_0, TCPK_1, TCSK_b)$ in any session other than the i-th session. In the i-th session, S acts as follows:

- In the i-th session, S acts just as the honest verifier $V(TCPK_0, TCPK_1, TCSK_b)$ does until he receives the Stage-2 message (α_0, α_1) from P^*.
- $k := 1$.
- While $k \leq 2$ do:
  - Uniformly selects $r_V \xleftarrow{R} \{0,1\}^n$ and sends r_V to P^* as the Stage-3 message.
  - Acts accordingly further by running $V(TCPK_0, TCPK_1, TCSK_b)$ until receiving the last Stage 4.3 message from P^*. Denote by $(\hat{a}, p_1)$ the Stage 4.1 message from P^*.
  - After receiving the Stage 4.3 message from P^*, S rewinds P^* to the point that P^* just sent Stage 4.1 message and sends back a new random Stage 4.2 message to P^*. Such a rewinding is called the *knowledge rewinding*.
  - Runs P^* further from the above knowledge rewinding point until receiving back again a Stage-4.3 message.
  - $k := k+1$. This means that S will rewind P^* to the point that P^* just sent the Stage-2 message (α_0, α_1) and send back a new random Stage-3 message. Such a rewinding is called the *major rewinding*.

Figure 4. The algorithm $S^{P^*}(1^n, TCPK)$

Firstly, we note that conditioned on $x_i \notin L$ the first case above will not appear in either the interactions prior to the major rewinding or the interactions posterior to the major rewinding. For the second case above, since b is chosen randomly in $\{0,1\}$, and the Σ_{OR} protocol used in Stage-1 is perfect witness indistinguishable, conditioned on $x_i \notin L$ and P^* always succeeds in completing its interactions with S, the probability of $j = 1 - b$ in both the interactions prior to the major rewinding and the interactions posterior to the major rewinding is $\frac{1}{4}$. Note that $TCPK_{1-b} = TCPK$ and from two different decommitments of α_{1-b} the corresponding secret-key $TCSK$ can be easily extracted.

Since P^* cannot distinguish whether he is interacting with honest verifier or S, suppose the protocol does not satisfy concurrent soundness then in the concurrent interactions between the (s,t)-concurrent malicious P^* and S, with the same non-negligible probability $p(n)$ there exists an k, $1 \leqslant k \leqslant s(n)$, such that S outputs "accept x_k" while $x_k \notin L$. Then since i is uniformly selected by S from $\{1, 2, \cdots, s(n)\}$, we conclude that with probability at least $\frac{(p(n))^4}{4s(n)}$, S gets either a witness w such that $(x_k, w) \in R_L$ or two different decommitments to α_{1-b} from which the secret-key $TCSK_{1-b}$ can be easily extracted. This contradicts to the assumption that $x_k \notin L$ and discrete logarithm is hard. Thus the protocol $< P, L >$ is concurrently sound in the BPK model.

That the system is an argument of knowledge is immediate from the extraction procedure of S. □

4.3 4-Round Practical Construction

We remark that the 6-round practical protocol above can be easily improved into a 4-round protocol. Specifically, for $x \in L$ that L admits Σ-protocols, the 4-round practical protocol is the following: I

In Stage-1 the verifier uses Σ_{OR}-protocol to prove that he knows one of secret-key with respect to the public-key pair.

In Stage-2, the prover uses Σ_{OR}-protocol to prove that he knows either a witness to $x \in L$ or one of secret-key with respect to the public-key pair.

The first and the second message of Stage-2 can be combined into Stage-1, and so the above protocol can be implemented in 4-round.

Furthermore, and more importantly, as we shall see in next subsection the 4-round practical protocol can be based on any one-way function that admits Σ-protocols by using the techniques presented in next section.

5 Improvements

5.1 4-round OWF-based CZK-CS

Now, we present the general construction that is a 4-round (that is optimal unless the language considered is trivial) black-box concurrent zero-knowledge argument of knowledge for $\mathcal{NP}$ with concurrent soundness in the BPK model under the minimal hardness assumption of one-way functions [2]. As usual, the general construction goes through $\mathcal{NP}$-reductions. The general protocol is depicted in Figure 5 (page 22)[3].

[2] This general construction is suggested by Lindell in January 2004.

[3] The general protocol is actually the Feige-Shamir constant-round zero-knowledge protocol for $\mathcal{NP}$ [40] with a bare public-key model added. But, the version of Feige-Shamir protocol used in our work is the one appearing in Feige's Ph.D. thesis [37] that is actually different to the one appearing in CRYPTO'89. This general construction is suggested by Yehuda Lindell although he declined the coauthorship of this work. We remark that both the practical construction and the above 6-round general protocol are developed independently of the Ph.D. thesis version of Feige-Shamir protocol.

The general protocol $\langle P, V \rangle$
Key Generation. Let f be a one-way function. For a security parameter n, each verifier randomly selects two elements x_1, x_2 from $\{0,1\}^n$, computes $y_1 = f(x_1)$ and $y_2 = f(x_2)$, publishes (y_1, y_2) as its public-key. For its secret-key, the verifier randomly selects a bit $b \stackrel{R}{\leftarrow} \{0,1\}$ and keeps x_b in secret as its secret-key while discards x_{1-b}.
Common input. An element $x \in L \cap \{0,1\}^n$. Denote by R_L the corresponding $\mathcal{NP}$-relation for L. **P private input.** An $\mathcal{NP}$-witness w for $x \in L$.
Stage 1. V uses a 3-round public-coin witness indistinguishability proof of knowledge (WIPOK) system for $\mathcal{NP}$ to prove that it knows a preimage to one of $y1, y2$. The witness used by V in this stage is x_b. **Stage 2.** P uses a 3-round public-coin WIPOK system for $\mathcal{NP}$ to prove either that it knows w s.t. $(x, w) \in R$ or that it knows a preimages to one of $y1, y2$. The witness used by P in this stage is w.

Figure 5. The general construction of zero-knowledge with concurrent player security for $\mathcal{NP}$ in the BPK model.

The protocol depicted in Figure 5 runs in 6-round, but it can be reduced into 4-round that is optimal according to the lower-bound proved in [58] on zero-knowledge with concurrent soundness in the BPK model. Note that the hardness assumption assumed in the general construction, i. e. the existence of one-way functions, is also minimal. But the general protocol goes through general $\mathcal{NP}$-reductions in both Stage-1 and Stage-2 and so it is not a practical solution.

Theorem 5.1 *Assuming the existence of one-way functions, the protocol $\langle P, V \rangle$ depicted in Figure 5 is a 4-round (that is optimal) black-box concurrent zero-knowledge argument of knowledge for $\mathcal{NP}$ that enjoys concurrent soundness in the BPK model.*

Proof (sketch).

(1) completeness.

If $x \in L$ then P can always complete the proof and V accepts it.

(2) Black-box concurrent zero-knowledge.

For any adversary V^* described in Section 4.1.3, the simulator S works in at most $s(n) + 1$ phases with black-box access to V^*, where $s(n)$ is the number of public-keys registered by V^* in the public-key file. In each phase, S either successfully gets a simulated transcript or "breaks" a new public-key in the sense that S can extract the corresponding secret-key (this is achieved by rewinding V^* to get two different accepting conversations of Stage-1 w. r. t. the same first message of Stage 1). Once a public-key (y_0, y_1) is broken (that is, S learns x_b), then in any session with respected to this broken public-key S works as follows: S runs accordingly just as an honest prover in Stage-1 but in Stage-2 he uses x_b as the witness to give the WIPOK.

Since S runs in at most $s(n)+1$ phases, and during each phase S also works in expected polynomial time, it is easy to see that S runs in expected polynomial time in toto.

The only difference between the simulated transcript generated by S and the real transcript of concurrent interactions between V^* and honest prover instances is that in the simulated transcript S uses the corresponding secret-key as the witness in Stage-2 of each session with respect to a broken public-key, while in the real transcript an honest prover uses the real $\mathcal{NP}$-witness of R_L as the witness in Stage-2 of each session. By the fact that witness indistinguishability is concurrently composable, using standard hybrid techniques it is easy to see that the simulated transcript is computational indistinguishable from the real transcript.

(3) Concurrent soundness and argument of knowledge.

We first note that a computational power unbounded prover can easily convince the verifier of a false statement since he can extract the secret-keys if his computational power is unbounded. Hence the protocol $\langle P, V\rangle$ depicted in Figure 6 constitutes an argument system rather than a proof system.

Suppose the protocol $\langle P, V\rangle$ does not satisfy concurrent soundness in the BPK model. According to the definition of concurrent soundness in the BPK model (described in Section 4.1.2), this means that in a concurrent attack issued by an (s,t)-concurrent malicious prover P^* against an honest verifier V with public-key (y_0, y_1) and secret-key x_b for $b \xleftarrow{R} \{0,1\}$, with non-negligible probability $p(n)$ there exists a k, $1 \leqslant k \leqslant s(n)$, such that V outputs "accept x_k" while $x_k \notin L$. Then we will construct an algorithm S that takes a value y as input and outputs either a witness for $x_k \in L$ or a preimage x to y such that $y = f(x)$ with probability at least $\frac{(p(n))^4}{4s(n)}$ in polynomial-time, which breaks either the assumption that $x_k \notin L$ or the hardness assumption of one-way functions.

The algorithm S is depicted in Figure 6 (page 23). For future reference convenience, we denote by Stage-2.1, 2.2, 2.3 the first, second, third message of Stage-2 respectively, where Stage-2.2 message is assumed to be a random string in $\{0,1\}^n$.

The algorithm $S^{P^*}(1^n, y)$

Compute $y' = f(x')$ for $x' \xleftarrow{R} \{0,1\}^n$.
$b \xleftarrow{R} \{0,1\}$.
Set y_b be y', x_b be x' and y_{1-b} be y. S publishes (y_0, y_1) as its public-key and keeps $x_b = x'$ in secret as its secret-key.

$i \xleftarrow{R} \{1, 2, \cdots, s(n)\}$.

S runs P^* and acts accordingly by running $V(y_0, y_1, x_b)$ in any session other than the i-th session. In the i-th session, S acts as follows:

- In the i-th session, S acts just as the honest verifier $V(y_0, y_1, y_b)$ until he receives the Stage-2.3 message from P^*.
- S rewinds P^* to the point that P^* just sent Stage-2.1 message and sends back a new random Stage-2.2 message to P^*.
- Runs P^* further from the above rewinding point until receiving back again a Stage-4.3 message.

Figure 6. The algorithm $S^{P^*}(1^n, y)$

According to the proof of knowledge property of Stage-2, conditioned on P^* always succeeds in completing its interactions with S, S gets either a witness for $x_i \in L$ or a preimage to one of (y_0, y_1). Furthermore, conditioned on $x_i \notin L$, according to the witness indistinguishability property of Stage-1, S will get a preimage to $y_{1-b} = y$ with probability one half except for negligibly small gaps.

Since we assume that the protocol depicted in Figure 5 does not satisfy concurrent soundness, it means that in the concurrent interactions between the (s,t)-concurrent malicious P^* and S, with non-negligible probability $p(n)$ there exists an k, $1 \leqslant k \leqslant s(n)$, such that S outputs "accept x_k" while $x_k \notin L$. Then since i is uniformly selected by S from $\{1, 2, \cdots, s(n)\}$, we conclude that with probability at least $\frac{(p(n))^2}{2s(n)}$ (except for negligibly small gaps), S gets either a witness w such that $(x_k, w) \in R_L$ or a preimage to $y_{1-b} = y$, which contradicts to the assumption that $x_k \notin L$ and the one-wayness of f. Thus the protocol $< P, L >$ is concurrently sound in the BPK model.

That the system is an argument of knowledge is immediate from the extraction procedure of S. □

5.2 How to Get rZK with Concurrent Soundness in the BPK Model

We remark that our 4-round (both general and practical) CZK-CS protocols can be easily modified into rZK-CS protocols by using the complexity leveraging technique introduced in [13] [4].

Specifically, the modifications are the following:

Modifications of V:

1. On the top of the 4-round CZK-CS, V firstly commits to a random string r_V of length n by using trapdoor commitments (specifically, using the DLP-based trapdoor commitment scheme in practical rZK-CS, and using the Feige-Shamir OWF-based trapdoor commitment scheme in general rZK-CS). Note that for trapdoor commitments the prover needs to firstly send a trapdoor public-key to V.

2. In the Stage-2 of the 4-round CZK-CS which is itself a 3-round public-coin WI, the second message of it (which is assumed to be a random string sent by V) is replaced by r_V. That is, V just decommits the trapdoor commitment in the second round of Stage-2.

Modifications of P:

All randomness used by P is computed by applying a PRF on the transcript up to know.

Complexity-Leverage Used

Note that the public-key of V includes a pair of y_0, y_1 which are in the range of the OWF f on inputs of length n. We let f has a larger security parameter than the security parameter of the trapdoor commitment scheme. Specifically, suppose the system security parameter is K which is also the security parameter of f, we let the security parameter of the trapdoor commitment is $k = K^\epsilon$, $\epsilon > 0$. We also assumes that the one-wayness of f against circuits of size 2^{K^ϵ}. This guarantee that we can use time 2^k to find the corresponding secret-key of the trapdoor commitment public-key but are still infeasible to break the one-wayness of f.

Comment on Round-Complexity

In above description, the modified rZK-CS protocol runs in 5-round. But if we allow provers also can publish public-keys in the public-key file, then the trapdoor commitment public-key can be deposited before the interactions take place. Note that in normal BPK model, only the verifiers publish public-keys. But, the BPK model does not exclude the case that provers also publish public-keys. In this general BPK model, our rZK-CS protocols actually run in 4-round that is optimal. Note that rZK-RS does not exist even in the BPK model.

5.2.1 Proof Outlines

The standard proof techniques for rZK presented in [13] can be directly applied here to show that the modified protocol is rZK.

For concurrent soundness, we remark that by using complexity-leveraging techniques, the proof procedure is almost the same of concurrent soundness of the original 4-round CZK-CS and is thus much simpler than that presented in [13].

[4] We remark that before we made this observation, we were informed from Di Crescenzo and Ivan Visconti that they had achieved rZK-CS protocol in the BPK model under superpolynomial hardness assumptions in a submission to CRYPTO2004. The work of Di Crescenzo et al. on rZK-CS is to appear in CRYPTO04. But, we stress that we make this observation without any details of the work of Di Crescenzo et al. on rZK-CS. Actually, we do not know any details of that work up to now. Our rZK-CS protocols are the minor modification of our CZK-CS, and although our protocols are under subexponential hardness assumption but it may be the most practical ones. Since we do not know any details of the work of Di Crescenzo et al, we cannot give comparisons between their protocols and our protocols at the current stage.

Specifically, using time 2^k the honest verifier can extracts the trapdoor commitment secret-key and thus can decommit in Stage-2 at its wish. This means that once the honest verifier get the trapdoor commitment secret-key, then the same proof procedure can be directly used here to show concurrent soundness of the modified protocol. We remark that in the sequential soundness proof of [13, 58], the simulator also needs to generate simulated WIPOK which also takes time exponentially in k by using complexity-leveraging. But in our proof, the simulator does not need complexity-leveraging in this simulation stage and so is simpler.

5.3 Practical rZK with Secret-Keys in Preprocessing Model

Zero-knowledge with secret-keys is recently studied by Cramer and Damgard in TCC04. The philosophy of ZK with secret-keys is that what can get from honest prover can also be computed by the verifier himself from his secret-key corresponding the public-key.

Our practical CZK-CS can be also modified into practical rZK with registered public-keys.

5.3.1 4-round practical rZK with verifiable public-keys

For any OWF that admits Σ-protocol, e. g. DLP, RSA et al, each verifier publishes a public-key such that the existence of the corresponding secret-key can be publicly verified. That is, in this model, the verifier does not need to give WIPOK for the knowledge of such secret-keys. In practice, the existence of secret-keys can be verified by an authority like a CA, or the verifier ZKPOK to CA the existence of secret-keys. After such ZKPOKS, the CA gives a certificate that the secret-key exists.

The prover also has a public-key for a trapdoor commitment scheme. For a language that admits Σ-protocol, the protocol works as follows:

Stage-1. 1. V sends $TC(r_l)$ for a random string r_l to P.

Stage-2: Using the Σ_{OR}-protocol, P proves that he knows either a witness for the common input or the secret-key of the public-key of the verifier. We remark that the second message of the Σ_{OR}-protocol (which is assumed to be random string sent from V to P) is the random string r_l committed in Stage-1.

By using standard complexity-leveraging techniques, it is easy to see that the above protocol is a practical 4-round rZK without going NP reductions in the preprocessing model defined in [13]. Furthermore, this protocol can be further improved into a 3-round rZK with resettable soundness in the preprocessing model. Specifically, the Stage-1 of above protocol is removed. The second message of Σ_{OR} is got by V by applying a VRF on the first message of Σ_{OR}, the common input and the registered public-keys. But the practical property of such a 3-round rZK with resettable soundness relies on the practical property of VRF that still cannot be viewed as very practical up to now.

5.3.2 Practical NIZK with random oracles in the preprocessing model

Specifically, if we work in the random oracle model then the second message of the above $\Sigma_{OR}\Sigma_{OR}$-protocol can be got by requesting the random oracle in order to get a non-interactive ZK in the preprocessing model. As ususal, in practice, the random oracle is replaced by using Hash functions. We think such a very simple but very practical NIZK in the preprocessing model may be very attractive to industry in practice.

How to Prevent Man-in-the-Middle Attacks: a 2-round practical ZK ID scheme in the random oracle model with preprocessing. Note that above practical NIZK in the random oracle model with preprocessing does not secure against man-in-the-middle attack. Specifically, a MIM adversary can relay the NIZK message to impersonate the honest prover. To prevent the MIM attacks, one way is to include a time-tag in the NIZK message such that the random get from the random oracle is got not only on the common input, the public-key, and the first message of Σ_{OR} but also on the

time tag; Another way is that we can let the verifier firstly send a random string r_V to P, and when P prepares the NIZK in the second round, he sends r_V, the common input, all public-keys(of both the verifier and the prover), and the second message of Σ_{OR} to the random oracle to get the second message of Σ_{OR}.

We also remark that the 4-round CZK-CS based on any OWF that admits Σ-protocols can be DIRECTLY modified into a practical NIZK in the random oracle model. Specifically, we let $(f(x_1), f(x_2))$ be the common reference string. In this setting, the verifier does not any longer need to proves the knowledge of one secret-key by using Σ_{OR}. That is, the stage-1 is avoided. The second message in Stage-2 (the random challenge sent by the verifier to the prover) will be generated by the prover itself by using the random oracle. We believe that such practical NIZK in the random oracle model will be very attractive in application and we are planning to apply for a US patent for it.

6 Cryptographic Protocols with both Concurrent Player Security and Concurrent Channel Security

In this section, we present a major application of our CZK-CS protocol: both OWF-based general and DLP-based practical 5-round concurrent coin-tossing protocols in the BPK model that also implies constant-round ZK and commitments with both concurrent player security and concurrent channel security (concurrent non-malleability) in the BPK model. Note that previous concurrent non-malleability result only is achieved in the common random string model and coin-tossing is one of the first and more fundamental problem in the literature [12].

6.1 Definition of Concurrent Coin-Tossing

In this section we formally define concurrently secure coin-tossing, which we name it *concurrent coin-tossing* CCT. Note that, according to discussions presented in Introduction, a two party protocol $< L, R >$ is a concurrent coin-tossing protocol if it satisfies three kinds of concurrent security requirements (to be addressed in detail in the following subsections): the concurrent security of unauthenticated communication channels (concurrent non-malleability), the concurrent security of left-players, and the concurrent security of right-players. Accordingly, the approach to the definition of concurrent coin-tossing is as follows: we first define the concurrent non-malleability of coin-tossing, and then define the concurrent player security of coin-tossing, finally a coin-tossing protocol is defined to be concurrent coin-tossing if it satisfies both the concurrent non-malleability and the concurrent player security.

6.1.1 Model and basic terminology of concurrent non-malleability

The concept of non-malleability is introduced by Dolve, Dwork and Naor [33] to avoid "man-in-the-middle" attacks. In the normal "man-in-the-middle setting" of two-party protocol $< L, R >$, non-malleability refers to the *unauthenticated* channel security between two *honest* players, L (the left player) and R (the right player), in a *stand-alone* execution of the protocol. In this paper we consider the concurrent version of the "man-in-the-middle setting" of two-party protocol, denoted CMIM, where polynomially many concurrent executions of the same protocol $< L, R >$ take place in an asynchronous setting (say, on the Internet) and all the *unauthenticated* communication channels (among all the concurrently executing instances of $< L, R >$) are controlled by a probabilistic polynomial-time (PPT) adversary A. This means that the left players cannot directly communicate with the right players since all communication messages are done through the adversary. Thus, we can divide the polynomially many concurrent executions of the protocol $< L, R >$ in the CMIM setting into two parts: the left part of CMIM, in which the adversary plays the role of right players and concurrently interacts with

polynomially many left players; and the right part of CMIM, in which the adversary plays the role of left players and concurrently runs with polynomially many right players. We call an adversary $s(n)$-adversary if the adversary involves at most $s(n)$ concurrent sessions in each part of the CMIM setting, where n is a security parameter and $s(\cdot)$ is a positive polynomial. The adversary A (controlling the scheduling of messages in both parts of CMIM) can decide to simply relay the messages of each left player in the left part to the corresponding right player in the right part. But it can also decide to block, delay, divert, or change messages arbitrarily at its wish. The CMIM setting with a PPT $s(n)$-adversary can be depicted in Figure 7 (page 28) [5] :

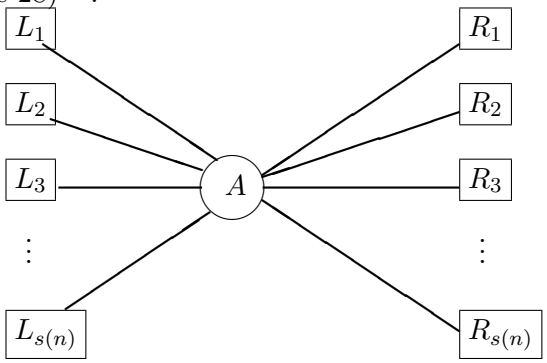

Figure 7. CMIM setting

In the CMIM setting, there are two extreme strategies that A can always use. One strategy is the *relaying* strategy in which the only thing A does is relay messages between each communicating pair $< L_i, R_i >$, $1 \leq i \leq s(n)$. In this case A is transparent. The other extreme strategy is the *blocking* strategy in which A plays its role in each session of one part of the CMIM setting completely independent of all the sessions running concurrently in another part of the CMIM setting. Since all the communication channels in the CMIM setting are unauthenticated, regardless of the protocol it is impossible to prevent the adversary from using one of the two strategies. Intuitively, the goal in designing protocols for the CMIM setting, is to design protocols that force A to use one of these two extreme strategies (or such that it could not be advantageous to A to use any other strategies).

Now, we are ready for a formal definition of concurrently non-malleable coin-tossing (CNMCT) in the CMIM setting. In the following definition we have assumed, without loss of generality, that the view of an adversary in the CMIM setting includes the outputs of all concurrent sessions.

Definition 6.1 (black-box concurrently non-malleable coin-tossing CNMCT) *Let* $\Pi =< L, R >$ *be a two-party protocol. We say that* Π *is a (black-box) concurrently non-malleable coin-tossing protocol in the CMIM setting if there exists a probabilistic (expected) polynomial-time algorithm* S *such that for any PPT* $s(n)$*-adversary* A *in the CMIM setting, the following holds:*

- *Simulatability. let* $S_L = \{S_L^{(1)}, S_L^{(2)}, \cdots, S_L^{s(n)}\}$ *and* $S_R = \{S_R^{(1)}, S_R^{(2)}, \cdots, S_R^{s(n)}\}$ *be two sets of random strings, each set containing* $s(n)$ *random strings. The output of* $S^A(S_L, S_R)$ *is computationally indistinguishable from the view of* A *in the real concurrent executions of* Π *in the CMIM setting. Here, the view of* A *is a random variable describing the random tape of* A*, all the messages sent by honest players, and the outputs of all concurrent sessions running in both parts of the CMIM setting. We remark that for protocols in the BPK model, the adversary's view also includes the public-key file.*

[5] In general, the number of concurrent sessions in the left part of CMIM is not necessarily equal to the number of concurrent sessions in the right part of CMIM, but here for simplicity and wlog we assume they are identical.

- *Strategy-restricted and predefinable randomness. Denote by* $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots, R_L^{s(n)}\}$ *the set of outputs (recorded in the output of* $S^A(S_L, S_R)$*) for all concurrent sessions between* S *and* A *in the left part of CMIM and denote by* $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots, R_R^{s(n)}\}$ *the set of outputs (recorded in the output of* S*) for all concurrent sessions between* S *and* A *in the right part of CMIM. Then, with overwhelming probability (except for negligibly small gaps),* $R_L = S_L$ *and for each* $R_R^{(i)}$ *in* R_R *(*$1 \leq i \leq s(n)$*) either* $R_R^{(i)} = S_R^{(i)}$ *or* $R_R^{(i)} \in R_L$*. Furthermore, with overwhelming probability for each* $R_L^{(j)}$ *in* R_L*,* $1 \leq j \leq s(n)$*,* $R_L^{(j)}$ *appears at most once in* R_R*.*

We remark that condition 3 in above definition of CNMCT is necessary to achieve concurrently non-malleable zero-knowledge without common random string (by combining a CNMCT protocol with the concurrently non-malleable NIZK in the common random string model [24]).

Motivations for concurrent non-malleability. Consider the concurrent executions of a zero-knowledge protocol in a distributed clients/server setting over Internet in which the client is implemented as the prover and the server is implemented as the verifier. We remark that this setting is widely used in practice, especially in E-commerce over Internet. At any time there may be numerous instances of the zero-knowledge protocol executing concurrently between many clients and the server. Since all the communication channels are unauthenticated, the channel security in suchlike settings is a real threat and concurrently non-malleable zero-knowledge protocols are really desirable in practice.

Another motivation example as stated in [21] is fair contract bidding, which can be implemented by having each participant commit to his bid first, after which commitments can be opened and the winner is determined. Consider the concurrent executions of the commitment scheme over Internet. We note that the non-malleable commitment schemes [30, 31, 41] in the common reference string model only guarantee that after seeing *one* commitment from the honest player an adversary is infeasible to generate another maliciously related commitment. Such a (stand-alone) non-malleability security does not suffice for secure fair contract bidding in the concurrent setting. For example, the security proofs of [30, 31] break down in the concurrent setting [21] and Damgard and Groth [21] show that there do exist commitment schemes that are stand-alone non-malleable but not concurrently non-malleable.

6.1.2 Concurrent player security

In this subsection, we present the concurrent left/right player security of a coin-tossing protocol $< L, R >$ in the concurrent setting.

The concurrent left player security essentially requires that for any PPT adversary that controls all the right-players and concurrently interacts with polynomially many instances of a left-player, if the left-player is honest then with overwhelming probability the outputs of all the concurrent sessions (between the adversary and polynomially many instances of the honest left-player) are completely independent random strings. Furthermore, in this paper we ask for a much more stronger left-player security in the knowledge-extraction sense that the adversary's view can be simulated by a simulator with oracle access to the adversary and the simulator can also uniformly set the simulated outputs (between the simulator and the adversary) at its wish.

Accordingly, the concurrent right player security essentially requires that for any PPT adversary that controls all the left-players and concurrently interacts with polynomially many instances of a right-player, if the right-player is honest then with overwhelming probability the outputs of all the concurrent sessions (between the adversary and polynomially many instances of the honest right-player) are completely independent random strings. Furthermore, the adversary's view can be simulated by a simulator with oracle access to the adversary and the simulator can also uniformly set the simulated outputs (between the simulator and the adversary) at its wish.

Definition 6.2 (black-box concurrent player security of coin-tossing) *Let $\Pi =< L, R >$ be a coin-tossing protocol. On a security parameter n, we call an adversary a $s(n)$-concurrent adversary if it is involved in at most $s(n)$ concurrent sessions between it and honest-player instances, where $s(n)$ is a positive polynomial in n. Let $R_S = \{r_S^{(1)}, r_S^{(2)}, \cdots, r_S^{(s(n))}\}$ be a set of $s(n)$ random strings of length n each. We say that Π satisfies black-box concurrent left-player (respectively, right-player) security if there exists a probabilistic (expected) polynomial-time algorithm S such that for any PPT $s(n)$-concurrent adversary A that plays the role of right-players (respectively, left-players) and concurrently interacts with polynomially many instances of the honest left-player (respectively, the right-player) in at most $s(n)$ sessions, the following holds:*

- *Simulatability. The output of $S^A(R_S)$ is computationally indistinguishable from the view of A in its real concurrent executions (that is a random variable describing the random tape of A, all the messages sent by honest-player instances, and the outputs of all concurrent sessions between A and the honest-player instances). We remark that for concurrent* right-player *security in the BPK model, the view of the adversary playing the role of left-player also includes the public-key file generated by honest right-players.*

- *Predefinable Ramdomness. With overwhelming probability, the simulated outputs of S are completely independent random strings and can be uniformly set by S at its wish. Specifically, denote by $R_A = \{r_A^{(1)}, r_A^{(2)}, \cdots, r_A^{(s(n))}\}$ the outputs (recorded in the output of $S^A(R_S)$) of all the concurrent sessions between S and A, then with overwhelming probability (except for at most negligible probabilities) $R_A = R_S$.*

6.1.3 Comments and observations on the definition of concurrent coin-tossing

Now we are ready to present the definition of concurrent coin-tossing.

Definition 6.3 (black-box concurrent coin-tossing CCT) *A two-party protocol $\Pi =< L, R >$ is called a black-box concurrent coin-tossing protocol if it is black-box concurrently non-malleable and satisfies both the black-box concurrent left-player security and the black-box concurrent right-player security.*

We first note that from the definition of concurrently non-malleable coin-tossing (Definition 2.1) it can be easily seen that a concurrently non-malleable coin-tossing protocol in the plain model (without any trusted third party or setup assumptions) is actually also a concurrent coin-tossing protocol in the plain model. That is, the definition of concurrent non-malleability of coin-tossing in the plain model actually also implies both the concurrent left-player security and the concurrent right-player security. The reason is that in the plain model the adversary controlling all the communication channels in the CMIM setting (as defined in Definition 2.1) can do what the adversary controlling either all the left-players or all the right-players (as defined in Definition 2.2) does. But this observation does not hold in the BPK model. In the BPK model, the left/right player security of a cryptographic two-party protocol may rely on the registered public-keys and the corresponding secret-keys that keep secret to the communication channels. For this reason, we treat concurrent non-malleability and concurrent player security respectively and adopt here the full version of concurrent coin-tossing definition that can be applied to both the plain model and computational models with registered public-keys or other setup assumptions.

Another observation is that the black-box concurrent left-player security of coin-tossing actually implies black-box concurrent zero-knowledge by composing a black-box concurrent coin-tossing protocol with a non-interactive zero-knowledge (NIZK) protocol in the common random-string model, where the left-player plays the role of zero-knowledge prover and the right-player plays the role of verifier. Since

any black-box concurrent zero-knowledge protocol for a language outside $\mathcal{BPP}$ in the plain model runs at least $\tilde{\Omega}(log\ n)$ rounds [14, 15], and since NIZK exists for $\mathcal{NP}$ in the common random-string model under the existence of trapdoor one-way permutations[39], so we conclude that assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$ black-box concurrent coin-tossing protocols run also at least $\tilde{\Omega}(log\ n)$ rounds in the plain model. In particular, it means that one cannot expect to achieve constant-round black-box concurrent coin-tossing protocols in the plain model. In this paper, we achieve constant-round black-box concurrent coin-tossing in the BPK model. By composing a constant-round black-box concurrent coin-tossing in the BPK model with an NIZK protocol in the common random string model, we get a constant-round black-box concurrent zero-knowledge argument of knowledge with concurrent soundness. Since any black-box zero-knowledge protocol with concurrent soundness in the BPK model for a language outside of $\mathcal{BPP}$ requires at least four rounds [58], we conclude that assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$ black-box concurrent coin-tossing protocols run also at least four rounds in the BPK model.

6.2 the OWF-based general construction

The OWF-based general constant-round concurrent coin-tossing (CCT) protocol in the BPK model is depicted in Figure 8 (page 43). Note that constant-round CCT does not exist in the plain model assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$ as we have argued in Section 2.

The General Concurrent Coin-Tossing $\Pi =< L, R >$

Key Generation. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a one-way function. On a security parameter n, the right-player R randomly selects x_0, x_1, x_2 from $\{0,1\}^n$, computes $y_0 = f(x_0)$, $y_1 = f(x_1)$ and $y_2 = f(x_2)$. Then by using the general $\mathcal{NP}$-reduction, the right-player transforms y_2 into a directed graph G. (y_0, y_1, G) is the public-key of the right player R. But for its secret-key, the right-player randomly selects a bit b from $\{0,1\}$, keeps x_b as its secret key while discards x_{1-b} and x_2.

Stage-1. Left player uniformly selects $r_l \overset{R}{\leftarrow} \{0,1\}^n$, and for a random string s_{r_l} computes $\alpha = \text{FSTC}(G, r_l, s_{r_l})$ by using the Feige-Shamir trapdoor commitment scheme FSTC with respect to G. Finally, the left player sends α to the right player.

Stage-2. The right player uniformly selects $r_r \overset{R}{\leftarrow} \{0,1\}^n$ and sends r_r to the left player.

Stage-3. The left player sends $r = r_l \oplus r_r$ to the right player.

Stage-4. Using the 4-round general concurrent zero-knowledge argument of knowledge (CZKAOK) with concurrent soundness (depicted in Figure ?), the left player proves that: he knows a string s' such that $\alpha = \text{FSTC}(G, r \oplus r_r, s')$. The witness used by the left player in Steps L4-7 is (r_l, s_{r_l}).

The result of the protocol is the string r. We will use the convention that if one of the parties aborts (or fails to provide a valid proof) then the other party determines the result of the protocol.

Figure 8. The general concurrent coin-tossing in the BPK model

The protocol depicted in Figure 6 runs in 7 rounds. But it can be reduced into 5 rounds by accordingly combining some rounds. Specifically, the Stage-1 and Stage-2 can be combined into the first and the second round of the underlying 4-round concurrent zero-knowledge protocol respectively.

Theorem 6.1 *Under the existence of one-way functions, the protocol* $\Pi =< L, R >$ *depicted in Figure 8 is a 5-round black-box concurrent coin-tossing protocol in the BPK model.*

Proof. According to the definition of black-box concurrent coin-tossing, we need to show that the protocol $\Pi =< L, R >$ satisfies both black-box concurrent channel security and black-box concurrent player security in the BPK model.

Black-box concurrent channel security (black-box concurrent non-malleability).

We first show that on a security parameter n, for any positive polynomial $s(n)$ and for any PPT $s(n)$-adversary A in the CMIM setting of the BPK model with respect to any public-key file of size $s(n)$ generated by honest right-players, the protocol $\Pi =< L, R >$ is black-box concurrently non-malleable. Specifically, for any PPT $s(n)$-adversary A in the CMIM setting of the BPK model with respect to any public-key file of size $s(n)$, let $S_L = \{S_L^{(1)}, S_L^{(2)}, \cdots, S_L^{s(n)}\}$ and $S_R = \{S_R^{(1)}, S_R^{(2)}, \cdots, S_R^{s(n)}\}$ be two sets of random strings, each set containing $s(n)$ random strings of length n each, we construct a simulator S that gets (S_L, S_R) as its input and generates in expected polynomial time (while oracle accessing to A) a simulated view satisfying the required properties according to the definition of concurrently non-malleable coin-tossing. For presentation convenience, besides the simulated key generation phase of S we describe the simulation procedure of $S^A(S_L, S_R)$ in two parts: the left part, in which S plays the role of honest left players and concurrently interacts with A; the right part, in which S plays the role of the honest right players and concurrently interacts with A. The algorithm $S^A(S_L, S_R)$ is depicted in Figure 9 (page 44).

First, we show that the simulator $S^A(S_L, S_R)$ runs in expected polynomial time. Note that whenever S rewinds A in the right simulation, the interactions between S and A in the left simulation are also rewound. But since the left simulation is straight-line simulatable and all the major rewindings in the right simulation are ordered , no simultaneous rewindings take place. Since A works in polynomial-time, there are also polynomially many major rewindings in the right simulation. And during each major rewinding S also runs in expected polynomial time by running the knowledge extractor of CZKAOK. We conclude that S works in expected polynomial time in total.

Below, we show that the simulation of S (depicted in Figure 7) does satisfy the required properties of concurrently non-malleable coin-tossing: simulatability, strategy-restricted and predefinable randomness.

(1) Simulatability.

According to the definition of concurrently non-malleable coin-tossing, this needs to show that the output of $S^A(S_L, S_R)$ is computationally indistinguishable from the view of A in real concurrent executions of the protocol $\Pi =< L, R >$ (depicted in Figure 6) in the CMIM setting of BPK model. Note that there are three differences between the view of A in $S^A(S_L, S_R)$ and the view of A in real concurrent executions.

1. For the i-th session in the left part of CMIM of the BPK model, $1 \leq i \leq s(n)$, in real execution, the honest left-player always sends $u \oplus \tilde{r}_r^{(i)}$ to A in Stage-3, where u is the random string committed by the honest left-player in Stage-1 of that session and $\tilde{r}_r^{(i)}$ is the Stage-2 message (sent by A) of that session; But in $S^A(S_L, S_R)$, S always sends $S_L^{(i)}$ in Stage-3.
2. For the i-th session in the left part of CMIM of the BPK model with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i, j \leq s(n)$, in real execution, the witness used by the honest left-player in Stage-4 is the randomness used to commit to u in Stage-1; But in $S^A(S_L, S_R)$, the witness used by S is generated from the trapdoor $x_2^{(j)}$.

Generation of Simulated Public-Key File F.

For $i := 1$ to $s(n)$ do:
$x_0^{(i)} \xleftarrow{R} TCGen(1^n)$; $x_1^{(i)} \xleftarrow{R} TCGen(1^n)$; $x_2^{(i)} \xleftarrow{R} TCGen(1^n)$.
Compute $y_0^{(i)} = f(x_0^{(i)})$, $y_1^{(i)} = f(x_1^{(i)})$ and $y_2^{(i)} = f(x_2^{(i)})$ and reduce $y_2^{(i)}$ to a graph $G^{(i)}$.
$b \xleftarrow{R} \{0,1\}$.
Publish $(y_0^{(i)}, y_1^{(i)}, G^{(i)})$ as the i-th public-key in the simulated public-key file F and keeps $(x_b^{(i)}, x_2^{(i)})$ as the corresponding secret-key while discards $x_{1-b}^{(i)}$.

Straight-Line Left Simulation	**Rewinding-Ordered Right Simulation**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between S and A in the left part of CMIM with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i, j \leq s(n)$, S acts in the following way: In Stage-1, S uniformly selects $u \xleftarrow{\mathrm{R}} \{0,1\}^n$ and sends $\alpha = FSTC(G_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, S sends $S_L^{(i)}$ to A in Stage-3 (rather than sending back $u \oplus \tilde{r}_r^{(i)}$ as the honest left player does). Then, in Stage-4, using the concurrent zero-knowledge argument of knowledge (CZKAOK) protocol S proves that "α is the commitment to $S_L^{(i)} \oplus \tilde{r}_r^{(i)}$". The witness used by S in Stage-4 is generated from $x_2^{(j)}$ (or equivalently, a directed Hamiltonian cycle in $G^{(j)}$) that is the trapdoor of the Feige-Shamir trapdoor commitment scheme with respect to $G^{(j)}$.	$i := 1$ Running A until receiving the first Stage-1 message, denoted α_i, marks the session of α_i the i-th session. **Label:** Suppose the i-th session is with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$. While "$A$ does not stop and S does not abort" do • Uniformly selects $v \xleftarrow{\mathrm{R}} \{0,1\}^n$ and sends v to A as the Stage-2 message of the i-th session. • Runs A further and acts accordingly (just as the honest right-player does) in any session other than the i-th session. When running into the CZKAOK phase (Stage 3-4) of the i-th session, denoted by the Stage-3 message (from A) of the i-th session $\tilde{r}_l^{(i)}$, S uses the knowledge-extractor of CZKAOK protocol to extract the witness used by A in the CZKAOK phase of the i-th session. This is achieved by rewinding A and such rewindings are called *knowledge rewindings*. If the extracted value is not (t_i, s_i) such that $\alpha_i = \mathrm{FSTC}(G^{(j)}, t_i, s_i)$, where t_i is of length n, then S aborts. Otherwise, S does the following: 1. rewinds A to the point A just sent α_i and sends back $S_R^{(i)} \oplus t_i$ to A. Such a rewinding is called the i-th *major rewinding*. 2. Runs A further (from the i-th major rewinding point) until receiving a new Stage-1 message from A; sets $i := i+1$ and marks the new Stage-1 message α_i and the current session of α_i the i-th session. 3. Goto **Label**

We stress that at any point of the simulation, if A does not act accordingly or fails to provide a valid proof then S aborts and outputs the public-key file F and the transcript up to now.

Figure 9. The simulation of $S^A(S_L, S_R)$

Generation of Simulated Public-Key File F.
This stage of S_1^A is identical to that of $S^A(S_L, S_R)$.

Left Interactions	**Right Interactions**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between S_1 and A in the left part of CMIM with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i, j \leq s(n)$, S_1 acts in the following way: In Stage-1, S_1 uniformly selects $u \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and sends $\alpha = FSTC(G_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, S_1 randomly selects a string $r_l^{(i)}$ in $\{0,1\}^n$ and sends $r_l^{(i)}$ to A in Stage-3. Then, in Stage-4, using the concurrent zero-knowledge argument of knowledge (CZKAOK) protocol S_1 proves that "α is the commitment to $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$". The witness used by S in Stage-4 is generated from $x_2^{(j)}$ (or equivalently, a directed Hamiltonian cycle in $G^{(j)}$) that is the trapdoor of the Feige-Shamir trapdoor commitment scheme with respect to $G^{(j)}$.	The right interactions between A and S_1 just identical to the interactions between A and honest right-players in real concurrent executions of the protocol $\Pi =< L, R >$ in the right part of CMIM of the BPK model. Specifically, there are no longer major-rewindings or knowledge-rewindings in the right interactions between S_1 and A, and at Stage-2 of any session S_1 just sends a random string of length n to A.
We stress that at any point of the simulation, if A does not act accordingly or fails to provide a valid proof then S_1 aborts and outputs the public-key file F and the transcript up to now.	

Figure 10. The hybrid experiment S_1^A

3. For the right part of CMIM of the BPK model, in $S^A(S_L, S_R)$ S may abort due to *incorrect* knowledge-extraction during the i-th major rewinding, $1 \leq i \leq s(n)$. Specifically, during the i-th major rewinding the output of knowledge rewindings may be a preimage to either y_0 or y_1 rather than (t_i, s_i) such that $\alpha_i = \mathrm{FSTC}(G^{(j)}, t_i, s_i)$ which is the value that S expects. In this case, S will abort the simulation. But this problem does not occur in real concurrent executions between A and honest right-players. We remark that S may also abort due to failed knowledge-extraction in the sense that it extracts no value from the knowledge rewindings. But according to the argument of knowledge property of the underlying concurrent ZK protocol, failed knowledge-extraction makes only negligible distinguishability gaps and so below we only consider the difference caused by *incorrect* knowledge-extraction.

We firstly consider the following hybrid experiment S_1^A in which the simulated public-key generation stage and left interactions in S_1^A are identical to that in $S^A(S_L, S_R)$, but the right interactions in S_1^A are identical to the interactions between A and honest right-players in real concurrent executions of the protocol $\Pi =< L, R >$ in the right part of CMIM of the BPK model. For clarity, S_1^A is depicted in Figure 10 (page 34).

Since failed knowledge-extraction makes only negligible distinguishability gaps, it is easy to see that conditioned on S aborts due to incorrect knowledge-extraction with negligible probability in $S^A(S_L, S_R)$,

the output of S_1^A is computationally indistinguishable from the output of $S^A(S_L, S_R)$. Then, the indistinguishability between the output of S_1^A and the output of $S^A(S_L, S_R)$ is followed from the following lemma.

Lemma 6.1 *The probability that S will abort due to incorrect knowledge-extraction in the rewinding-ordered right simulation of $S^A(S_L, S_R)$ is negligible.*

Proof. Let $F' = \{y_1, y_2, \cdots, y_{s(n)}\}$ be a set of $s(n)$ values in the range of f on inputs of length n. Suppose S aborts due to incorrect knowledge-extraction with non-negligible probability, then we will construct a probabilistic algorithm E that takes F' as input and works in expected polynomial-time (with oracle access to A) to output the preimage of y_i for some i, $1 \leq i \leq s(n)$, which violates the one-wayness of f. Actually, as we shall see, $E^A(F')$ works just as $S^A(S_L, S_R)$ does but with a modified simulated public-file generation. Specifically, in $E^A(F')$ the simulated public-file is generated from F' rather than generated by itself from scratch. For clarity, the full description of the algorithm $E^A(F')$ is depicted in Figure 11 (page 36).

It is easy to see that the view of A in $E^A(F')$ is identical to that of A in $S^A(S_L, S_R)$. Then, if S aborts due to incorrect knowledge-extraction with non-negligible probability p in the rewinding-ordered right simulation of $S^A(S_L, S_R)$, then with the same probability E will also abort due to incorrect knowledge-extraction in $E^A(F')$. This means that with probability p E will get a preimage of either $y_0^{(i)}$ or $y_1^{(i)}$ for some i, $1 \leq i \leq s(n)$. But, according to the (non-uniform) witness indistinguishability property of WIPOK used by honest right-players, with probability close to $p/2$ the extracted value will be the preimage of $y_i = y_{1-b}^{(i)}$, which violates the one-wayness of f.

□

Now, we consider the second hybrid experiment S_2^A. The simulated public-key file generation stage and the right interactions are identical to those of S_1^A. In left interactions, in Stage-3 of any session with respect to a public-key $(y_0^{(i)}, y_1^{(i)}, G^{(i)}$, S_2 sends $u \oplus \tilde{r}_r^{(i)}$ to A just as the honest left-player does, where u is the random string committed by S_2 in Stage-1 and $\tilde{r}_r^{(i)}$ is the message sent by A in Stage-2. But, in Stage-4 the witness used by S is still generated from $x_2^{(j)}$ (or equivalently, a directed Hamiltonian cycle in $G^{(j)}$) that is the trapdoor of the Feige-Shamir trapdoor commitment scheme with respect to $G^{(j)}$. For clarity, the description of S_2^A is depicted in Figure 12 (page 37).

We first note that the only difference between the view of A in S_2^A and the view of A in real concurrent executions of the protocol $\Pi = \langle P, V \rangle$ in the CMIM setting of the BPK model is that: In real executions, in Stage-4 of any session the honest left-player always uses the randomness used in its Stage-1 commitment; In S_2^A, the witness used by S_2 in Stage-4 is generated from the commitment trapdoor. However, according to the WI property of Stage-4, this difference only makes negligible distinguish gap since otherwise using standard hybrid technique one can construct a (non-uniform) algorithm that works in probabilistic polynomial-time to break the WI property of Stage-4. Thus, we conclude that the view of A in S_2^A is computationally indistinguishable from the view of A in real concurrent executions.

Now, all left is to show that the output of S_2^A is indistinguishable from the output of S_1^A. Actually, the indistinguishability between the output of S_2^A and the output of S_1^A is directly followed from the following lemma.

Lemma 6.2 *Suppose the output of S_2^A is not indistinguishable from the output of S_1^A, then we can construct a $\mathcal{BPP}$ machine that breaks the computational-hiding property of the underlying Feige-Shamir trapdoor commitment scheme with non-negligible acceptance gap.*

Generation of Simulated Public-Key File F From F'.
For $i := 1$ to $s(n)$ do:
$x' \xleftarrow{R} \{0,1\}^n$; $x'' \xleftarrow{R} \{0,1\}^n$.
Compute $y' = f(x')$, $y'' = f(x'')$ and reduce y'' to a graph $G^{(i)}$.
$b \xleftarrow{R} \{0,1\}$.
Set $y_b^{(i)}$ be y', and $y_{1-b}^{(i)}$ be y_i.
Publish $(y_0^{(i)}, y_1^{(i)}, G^{(i)})$ as the i-th public-key in the simulated public-key file F and keeps (x', x'') in secret as the corresponding secret-key.

Straight-Line Left Simulation	**Rewinding-Ordered Right Simulation**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between E and A in the left part of CMIM with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \le i, j \le s(n)$, E acts in the following way: In Stage-1, E uniformly selects $u \xleftarrow{\mathrm{R}} \{0,1\}^n$ and sends $\alpha = FSTC(G_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, E randomly selects a string $r_l^{(i)}$ in $\{0,1\}^n$ and sends $r_l^{(i)}$ to A in Stage-3 (rather than sending back $u \oplus \tilde{r}_r^{(i)}$ as the honest left player does). Then, in Stage-4, using the concurrent zero-knowledge argument of knowledge (CZKAOK) protocol E proves that "α is the commitment to $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$". The witness used by E in Stage-4 is generated from $x_2^{(j)}$ (or equivalently, a directed Hamiltonian cycle in $G^{(j)}$) that is the trapdoor of the Feige-Shamir trapdoor commitment scheme with respect to $G^{(j)}$.	$i := 1$ Running A until receiving the first Stage-1 message, denoted α_i, marks the session of α_i the i-th session. **Label:** Suppose the i-th session is with respect to a public-key $(TCPK_0^{(j)}, TCPK_1^{(j)}, TCPK_2^{(j)})$. While "$A$ does not stop and S does not abort" do • Uniformly selects $v \xleftarrow{\mathrm{R}} \{0,1\}^n$ and sends v to A as the Stage-2 message of the i-th session. • Runs A further and acts accordingly in any session other than the i-th session. When running into the CZKAOK phase (Stage 3-4) of the i-th session, denoted by the Stage-3 message (from A) of the i-th session $\tilde{r}_l^{(i)}$, E uses the knowledge-extractor of CZKAOK protocol to extract the witness used by A in the CZKAOK phase of the i-th session. This is achieved by rewinding A and such rewindings are called *knowledge rewindings*. If the extracted value is (t_i, s_i) such that $\alpha_i = \mathrm{TCCom}(TCPK_2^{(j)}, t_i, s_i)$, where t_i is of length n, then E does the following: 1. rewinds A to the point A just sent α_i, randomly selects a string $r_r^{(i)}$ in $\{0,1\}^n$ sends back $r_r^{(i)} \oplus t_i$ to A. Such a rewinding is called the i-th *major rewinding*. 2. Runs A further (from the i-th major rewinding point) until receiving a new Stage-1 message from A; sets $i := i+1$ and marks the new Stage-1 message α_i and the current session of α_i the i-th session. 3. Goto **Label**
We stress that at any point of simulation, if A does not act accordingly or fails to provide a valid proof then E aborts and outputs the public-key file F and the transcript up to now.	

Figure 11. The algorithm $E^A(F')$

Generation of Simulated Public-Key File F.
This stage of S_2^A is identical to that of S_1^A.

Left Interactions	**Right Interactions**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between S_2 and A in the left part of CMIM with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i, j \leq s(n)$, S_2 acts in the following way: In Stage-1, S_2 uniformly selects $u \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and sends $\alpha = FSTC(G_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, S_2 sends back $r_l^{(i)} = u \oplus \tilde{r}_r^{(i)}$ $r_l^{(i)}$ to A in Stage-3 just as the honest left-player does. Then, in Stage-4, using the concurrent zero-knowledge argument of knowledge (CZKAOK) protocol S_2 proves that "α is the commitment to $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$". The witness used by S_2 in Stage-4 is generated from the trapdoor $x_2^{(j)}$ (or equivalently, a directed Hamiltonian cycle in $G^{(j)}$) rather than the randomness used to commit to u in Stage-1 as the honest left-player does.	The right interactions of S_2^A between A and S_2 just identical to that of S_1^A between A and S_1.
We stress that at any point of the simulation, if A does not act accordingly or fails to provide a valid proof then S_2 aborts and outputs the public-key file F and the transcript up to now.	

Figure 12. The hybrid experiment S_2^A

Proof. We consider the following $s(n)$ hybrid experiments $H_1^A, H_2^A, \cdots, H_{s(n)}^A$. For any i, $1 \leq i \leq s(n)$, the simulated public-key file generation stage and the right interactions of H_i^A identical to those of S_1^A and S_2^A. But in its left interactions between H_i and A, in the first i sessions (ordered by the time-step the first round of each session is played) H_i works just as S_1 does, but in the rest $s(n)-i$ sessions H_i works just as S_2 does.

Specifically, in Stage-4 of all sessions H_i uses the commitment trapdoor to generate the witness to be used in the CZKAOK protocol. But in the j-th session, $1 \leq j \leq i$, the CZKAOK protocol is with respect to a random string chosen by H_i in Stage-3 and in the j-th session, $i < j \leq s(n)$, the CZKAOK protocol is with respect to the outcome of coin-tossing $r_l^{(i)} = u \oplus \tilde{r}_r^{(i)}$, where u is the random string committed in Stage-1 and $\tilde{r}_r^{(i)}$ is the message sent by A in Stage-2.

Clearly, suppose the output of S_2^A is distinguishable from the output of S_1^A with non-negligible probability $p(n)$, then there must exist an i, $1 < i \leq s(n)$, such that there exists a (non-uniform) PPT distinguisher D' that can distinguish the output of H_i and the output of H_{i-1} with probability at least $\frac{p(n)}{s(n)}$ that is also non-negligible in n. However, below we transform D' into another $\mathcal{BPP}$ machine D that breaks the computational-hiding property of the underlying Feige-Shamir trapdoor commitment scheme with non-negligible acceptance gap $\frac{p(n)}{s(n)^2}$.

Let r_0, r_1 be two random strings in $\{0,1\}^n$, and $\alpha = FSTC(r_b)$ for $b \xleftarrow{R} \{0,1\}$. Now, D takes α and r_k ($k = 0$ or 1) as its inputs and its task is to distinguish whether or not α commits to r_k. To do this, $D(\alpha, r_k)$ first randomly "guesses" an i from $\{1, 2, \cdots, s(n)\}$ and then runs A to mimic the experiment H_i^A but with the following exception. In the i-th session of left interactions between D and A, D sends α to A in Stage-1, and after receive the Stage-2 message $\tilde{r}_r^{(i)}$ from A, D sends $r_k \oplus \tilde{r}_r^{(i)}$ as the Stage-3 message. Finally, D gives the interaction transcripts (between D and A) to the distinguisher D' and outputs what D' outputs.

Now, we consider the acceptance gap between $\Pr[D(\alpha, r_0) = 1]$ and $\Pr[D(\alpha, r_1) = 1]$. The key observation is that: on one hand, if $r_k = r_{1-b}$ then the Stage-3 message $r_k \oplus \tilde{r}_r^{(i)}$ is also a random string and so in this case the interactions between A and D identical to H_i^A; on the other hand, if $r_k = r_b$ then the Stage-3 message $r_k \oplus \tilde{r}_r^{(i)}$ is the real outcome of the coin-tossing and so in this case the interactions between A and D identical to H_{i-1}^A. Since we assume that there exists an i, $1 < i \leq s(n)$, such that D' can distinguish the output of H_i and the output of H_{i-1} with probability at least $\frac{p(n)}{s(n)}$, and also since D chooses i randomly from $\{1, 2, \cdots, s(n)\}$, we conclude that $|\Pr[D(\alpha, r_0) = 1] - \Pr[D(\alpha, r_1) = 1]| \geq \frac{p(n)}{s(n)^2}$. This means that D is a $\mathcal{BPP}$ machine to break the computational-hiding property of the underlying Feige-Shamir trapdoor commitment scheme. □

(2) Strategy-restricted and predefinable randomness.

Denote by $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots, R_L^{p(n)}\}$ the set of outputs recorded in the output S for all concurrent sessions of the left part of CMIM in BPK model and denote by $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots, R_R^{p(n)}\}$ the set of outputs recorded in the output of S for all concurrent sessions of the right part of CMIM in BPK model.

It is clear that for any i, $1 \leq i \leq p(n)$, $R_L^{(i)} = S_L^{(i)}$ that is a random string of length n. For any string $R_R^{(i)} \in R_R$, there are three cases:

Case 1. $R_R^{(i)} = S_R^{(i)}$ that is a random string of length n.

Case 2. $R_R^{(i)} \in R_L$

Case 3. Other cases.

Generation of Simulated Public-Key File F From F''.
For $i := 1$ to $s(n)$ do:
$x' \xleftarrow{R} \{0,1\}^n$.
Compute $y' = f(x')$
$b \xleftarrow{R} \{0,1\}$.
Set $y_b^{(i)}$ be y', $y_{1-b}^{(i)}$ be $y^{(i)}$, and reduce $y_2^{(i)}$ to an instance of DHC $G^{(i)}$.
Publish $(y_0^{(i)}, y_1^{(i)}, G^{(i)})$ as the i-th public-key in the simulated public-key file F and keeps $x_b^{(i)} = x'$ in secret as the corresponding secret-key.

Straight-Line Left Simulation	**Rewinding-Ordered Right Simulation**
In any session of the left interactions between $\hat{E}$ and A, $\hat{E}$ works just as the honest left-player does. Specifically, in Stage-3 of any session, $\hat{E}$ sends $u \oplus \tilde{r}_r^{(i)}$ to A, where u is the random string committed by $\hat{E}$ in Stage-1 of that session and $\tilde{r}_r^{(i)}$ is the Stage-2 message from A. In Stage-4, $\hat{E}$ uses the corresponding randomness used to commit to u in Stage-1 as the witness.	The right interactions between $\hat{E}$ and A are identical to the rewinding-ordered right simulation of $E^A(F')$.

We stress that at any point of simulation, if A does not act accordingly or fails to provide a valid proof then $\hat{E}$ aborts and outputs the public-key file F and the transcript up to now.

Figure 13. The algorithm $\hat{E}^A(F'')$

According to the definition of strategy-restricted and predefinable randomness, it needs to show that the probability that Case 3 occurs is negligible, and that with overwhelming probability each string in R_L can appear in R_R at most once.

We first show the probability that Case 3 occurs is negligible. Let $F'' = \{(y^{(1)}, y_2^{(1)}), (y^{(2)}, y_2^{(2)}), \cdots, (y^{(s(n))}, y_2^{(s(n))}),\}$, where for each i, $1 \leq i \leq s(n)$, $y^{(1)}$ and $y_2^{(1)}$) are in the range of the OWF f on inputs chosen randomly from $\{0,1\}^n$. Now, suppose Case 3 occurs with non-negligible probability (i. e. with non-negligible probability there exists an i such that the output of the i-th session in the right simulation of S with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$ is neither Case 1 nor Case 2, $1 \leq i, j \leq s(n)$), then we will construct a probabilistic algorithm $\hat{E}^A(F'')$ that works in (expected) polynomial-time to break the onewayness of f. $\hat{E}^A(F'')$ works like the $E^A(F')$ (depicted in Figure 9) but with the following exceptions: The simulated public-key file in $\hat{E}^A(F'')$ is generated from F'' and in the straight-line left simulation of $\hat{E}^A(F'')$, $\hat{E}$ works just as honest left-player do. For presentation clarity, $\hat{E}^A(F'')$ is depicted in Figure 13.

It is easy to see that conditioned that $\hat{E}$ aborts due to incorrect knowledge-extraction with negligible probability in the rewinding-ordered right simulation, the view of A in $\hat{E}^A(F'')$ is identical to that of A in real concurrent executions of the protocol $\Pi = \langle P, V \rangle$ in the CMIM setting of the BPK model. Actually, using the same proof technique of Lemma 6.1, it is easy to see that the probability that $\hat{E}$ aborts due to incorrect knowledge-extraction in the rewinding-ordered right simulation is negligible. Since as we have shown the view of A in $S^A(S_L, S_R)$ is indistinguishable from the view of A in real concurrent interactions,

we conclude that if with non-negligible probability there exists an i such that the output of the i-th session in the right simulation of S with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$ is neither Case 1 nor Case 2, then with also non-negligible probability (suppose this non-negligible probability is $p(n)$) there exists an i such that the output of the i-th session in the right simulation of $\hat{E}$ with respect to a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$ is neither Case 1 nor Case 2, where $1 \leq i, j \leq s(n)$. However, by randomly "guessing" i from $\{1, 2, \cdots, s(n)\}$ and by knowledge-rewinding A using the knowledge extractor of CZKAOK after the major-rewinding point of the i-th session, $\hat{E}$ will extract another different decommitment to the same Stage-1 message α_i of the i-th session in expected polynomial-time with non-negligible probability approximately $\frac{p(n)}{s(n)}$. Note that from two different decommitments to the same α_i (one is extracted before the major-rewinding point of the i-th session and another is extracted after the major-rewinding point of the i-th session) E will easily computes out a Hamiltonian cycle of G^j from which one can easily compute a preimage of $y_2^{(j)}$, which contradicts the one-wayness of f.

Now, all left is to show that with overwhelming probability each string in R_L can appear in R_R at most once. It is sufficient to show that with overwhelming probability R_R consists of distinct values. Suppose with non-negligible probability there exists an i such that $R_R^{(i)} = R_R^{(j)}$ for $j < i$, then by randomly "guessing" i from $\{1, 2, \cdots, s(n)\}$ and by knowledge-rewinding A after the major-rewinding point of the i-th session, $\hat{E}$ will extract another decommitment $R_R^{(j)} \oplus S_R^{(i)} \oplus t_i$ to the same Stage-1 message α_i of the i-th session in expected polynomial time with non-negligible probability, where t_i is the decommitment to α_i extracted before the major-rewinding point of the i-th session. Since $S_R^{(i)}$ is a truly random value, with overwhelming probability $R_R^{(j)} \oplus S_R^{(i)} \oplus t_i \neq t_i$. Again, from two different decommitments to the same α_i E can easily extract a preimage of $y_2^{(i)}$, which violates the one-wayness of f.

Black-box concurrent player security.

We now proceed to prove the black-box concurrent player security of the protocol $\Pi =< L, R >$ depicted in Figure 6.

(1) Black-box concurrent right-player security.

Recall that in the concurrent right-player security setting, a adversary plays the role of left-players and concurrently interacts with polynomially many instances of the right-player. But since left-players are not involved in the key-generation stage of the BPK model, the adversary playing the role of left-players keeps no secret information. It means that for any adversary A in the concurrent right-player security setting of the BPK model there exists another adversary B in the CMIM setting of the BPK model, such that what the adversary A can do in its concurrent attacks against honest right-players can also be done by B in the right part of CMIM in the BPK model. Furthermore, B can do that independently of the concurrent executions in the left part of CMIM. This means that black-box concurrent channel security in the BPK model actually implies black-box concurrent right-player security in the BPK model. Note that we have proved that the protocol $\Pi =< L, R >$ does satisfy black-box concurrent channel security in the BPK model, so it also satisfies black-box concurrent right-player security in the BPK model.

(2) Black-box concurrent left-player security.

For black-box concurrent left-player security in the BPK model, we first observe that black-box concurrent channel security in the BPK model does not imply black-box concurrent left-player security in the BPK model any longer. The reason is that in the concurrent left-player security setting of the BPK model , an adversary A firstly generates and publishes a public-key file and then concurrently interacts with polynomially many instances of the honest left-player with respect to the published public-file.

Thus, the adversary keeps secret information (the corresponding secret-keys) and his behavior may also depend on such secret information. An algorithm without the knowledge of the secret information (like adversaries in the CMIM setting of the BPK model) may not do what the adversary A can do.

According to the definition of black-box concurrent left-player security, we need to construct a probabilistic (expected) polynomial-time simulator S such that for any PPT $s(n)$-concurrent adversary A that plays the role of right-players and concurrently interacts with many instances of the honest left-player in at most $s(n)$ sessions, the output of $S^A(R_S)$ satisfies the properties of simulatability and predefinable randomness, where $R_S = \{r_S^{(1)}, r_S^{(2)}, \cdots, r_S^{(s(n))}\}$ be a set of $s(n)$ random strings of length n each. Without loss of generality, we also suppose A generates and publishes a public-file that contains $s(n)$ public-keys of the form $(y_0^{(i)}, y_1^{(i)}, G^{(i)})$, $1 \leq i \leq s(n)$. Actually, as we shall see, the proof of black-box concurrent left-player security is very similar to the proof of black-box concurrent zero-knowledge for the general CZK-CS protocol ?. Specifically, just like the black-box zero-knowledge simulator presented in the proof of Theorem ?, $S^A(R_S)$ also works in at most $s(n)+1$ phases and in each phase, S either successfully gets a simulated transcript or "breaks" a new public-key in the sense that S can extract the corresponding secret-key $x_b^{(j)}$ $(1 \leq j \leq s(n))$. But, we remark that in the i-th session of each phase, $1 \leq i \leq s(n)$, S sends to A $r_S^{(i)}$ rather than $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$ in Stage-3, where $r_l^{(i)}$ is the random value committed by S in Stage-1 of the i-th session and $\tilde{r}_r^{(i)}$ is the value sent by A to S in Stage-2. And once a public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i \leq s(n)$, is broken (that is, S learns $x_b^{(j)}$), then S uses the witness generated from $x_b^{(j)}$ in Stage-4.

Since S runs in at most $s(n)+1$ phases, and during each phase S also works in expected polynomial time, it is easy to see that S runs in expected polynomial time in toto.

There are two differences between the output of $S^A(R_S)$ and the view of A in real interactions with honest left-player instances:

1. In the simulated transcript generated by $S^A(R_S)$, S sends to A $r_S^{(i)}$ (rather than $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$ as the honest left-player does) in Stage-3 of the i-th session , $1 \leq i \leq s(n)$, where $r_l^{(i)}$ is the random value committed by S in Stage-1 of the i-th session and $\tilde{r}_r^{(i)}$ is the value sent by A to S in Stage-2.

2. In the i-th session of real interactions, the honest left-player uses the corresponding randomness (used to commit to $r_l^{(i)}$ in Stage-1) as the witness in Stage-4. But in the simulated transcript, in any session with respect to a (broken) public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i \leq s(n)$, the witness used by S in Stage-4 is generated from the extracted secret-key $x_b^{(j)}$.

To show the indistinguishability between the output of $S^A(R_S)$ and the view of A in real interactions with honest left-player instances, we consider the following mental hybrid experiment H^A in which a PPT algorithm H just works as S does but with the following exceptions: During any phase of the at most $s(n)+1$ phases, in the i-th session with respect to a broken public-key $(y_0^{(j)}, y_1^{(j)}, G^{(j)})$, $1 \leq i, j \leq s(n)$, H sends $r_l^{(i)} \oplus \tilde{r}_r^{(i)}$ (as the honest left-player does) in Stage-3, where $r_l^{(i)}$ is the random value committed by H in Stage-1 of the i-th session and $\tilde{r}_r^{(i)}$ is the value sent by A to H in Stage-2. But in Stage-4, the witness used by H is generated from the extracted secret-key $x_b^{(j)}$.

The indistinguishability between the output of H^A and the view of A in real concurrent interactions with honest left-player instances is followed from the (concurrent) witness indistinguishability property of Stage-4. The indistinguishability between the output of $S^A(R_S)$ and the output of H^A is followed from the fact that if they are not indistinguishable then using the similar proof procedure of Lemma 6.2 one can construct a $\mathcal{BPP}$ machine that breaks the computational-hiding property of the underlying Feige-Shamir trapdoor commitment scheme with non-negligible acceptance gap.

□

Comparisons of our result with the works of Barak and Lindell [2, 53].

Firstly, I remark that the structure of our coin-tossing follows that of Barak's stand-alone non-malleable coin-tossing [2] that in turn follows the structure of Lindell's parallel coin-tossing [53].

There are two major differences between the protocol construction of our work and that of [53, 2]. One is that in the works of [53, 2], there is an additional POK phase between Stage-1 and Stage-2 in which the left player proves the knowledge of the value he committed to in Stage-1. This additional POK phase is needed in [53] in order to simulate the view of a malicious (aborting) left player in the secure two-party computation setting. In [2] the additional POK phase is to avoid simultaneous rewindings. This additional POK phase is avoided in our construction. The reason is: on one hand, non-malleability refers to unauthenticated channel security and all players are assumed *honest*; on the other hand, in the simulation of S no simultaneous rewinding occurs since the left simulation is straight-line simulatable. Another major difference is that in [53, 2] the left player uses a *perfect-binding* commitment scheme but in our work the left player uses a *computational-binding* trapdoor commitment scheme. We remark that the trapdoor commitments play a critical role in our construction.

We further make some comparisons of our result with the work of Barak [2]. The work of [2] is the first constant-round *stand-alone* non-malleability result in the plain model (without any trusted dealer or setup assumption). But the construction and security analysis of the protocol [2] are much involved. The protocol of [2] follows the outline of the protocol of [53] and additionally uses evasive set families and ZK universal arguments [3]. It is based on sub-exponential hardness assumptions and runs at least 10 rounds since the known ZK universal arguments runs in 10 rounds. The security analysis of [2] involves diagonalization, non-black-box simulation[1, 5], and the Richardson-Killian transformation [65] techniques and treats the synchronizing and non-synchronizing adversaries separately. In comparison, our work deals with *concurrent* non-malleability in the BPK model. Our protocol does not assume any sub-exponential hardness assumption and can be implemented in 5 rounds. The security analysis also does not employ much involved techniques. Our result can be viewed as another evidence to the powerfulness of the BPK model.

6.3 The DLP-based Practical Construction

The DLP-based practical construction is depicted in Figure 14.

The Practical Concurrent Coin-Tossing $\Pi =< L, R >$

Key Generation. For a security parameter n, let $(TCPK_0, TCSK_0) \stackrel{R}{\leftarrow} \text{TCGen}(1^n)$, $(TCPK_1, TCSK_1) \stackrel{R}{\leftarrow} \text{TCGen}(1^n)$ and $(TCPK_2, TCSK_2) \stackrel{R}{\leftarrow} \text{TCGen}(1^n)$. $(TCPK_0, TCPK_1, TCPK_2)$ is the public-key of the right player R. But for its secret-key, the right-player randomly selects a bit b from $\{0,1\}$, keeps $TCSK_b$ as its secret key while discards $TCSK_{1-b}$ and $TCSK_2$.

Stage-1. Left player uniformly selects $r_l \stackrel{R}{\leftarrow} \{0,1\}^n$ and, for a random string s_{r_l}, computes $\alpha = \text{TCCom}(TCPK_2, r_l, s_{r_l})$ using the trapdoor commitment scheme TC with respect to $TCPK_2$. Finally, the left player sends α to the right player.

Stage-2. The right player uniformly selects $r_r \stackrel{R}{\leftarrow} \{0,1\}^n$ and sends r_r to the left player.

Stage-3. The left player sends $r = r_l \oplus r_r$ to the right player.

Stage-4. Using the 6-round practical concurrent zero-knowledge argument of knowledge (CZKAOK) with both concurrent soundness and concurrent witness extraction (depicted in Figure 2), the left player proves that: he knows a string s' such that $\alpha = \text{TCCom}(TCPK_2, r \oplus r_r, s')$. The witness used by the left player in Steps L4-7 is (r_l, s_{r_l}).

The result of the protocol is the string r. We will use the convention that if one of the parties aborts (or fails to provide a valid proof) then the other party determines the result of the protocol.

Figure 14. The practical concurrent coin-tossing in the BPK model

Theorem 6.2 *Under discrete logarithm assumption, the protocol* $\Pi =< L, R >$ *depicted in Figure 14 is a practical black-box concurrent coin-tossing protocol in the BPK model without going through general* $\mathcal{NP}$*-reductions .*

Proof. According to the definition of black-box concurrent coin-tossing, we need to show that the protocol $\Pi =< L, R >$ satisfies both black-box concurrent channel security and black-box concurrent player security in the BPK model.

Black-box concurrent channel security (black-box concurrent non-malleability).

We first show that on a security parameter n, for any positive polynomial $s(n)$ and for any PPT $s(n)$-adversary A in the CMIM setting of the BPK model with respect to any public-key file of size $s(n)$ generated by honest right-players, the protocol $\Pi =< L, R >$ is black-box concurrently non-malleable. Specifically, for any PPT $s(n)$-adversary A in the CMIM setting of the BPK model with respect to any public-key file of size $s(n)$, let $S_L = \{S_L^{(1)}, S_L^{(2)}, \cdots, S_L^{s(n)}\}$ and $S_R = \{S_R^{(1)}, S_R^{(2)}, \cdots, S_R^{s(n)}\}$ be two sets of random strings, each set containing $s(n)$ random strings of length n each, we construct a simulator S that gets (S_L, S_R) as its input and generates in expected polynomial time (while oracle accessing to A) a simulated view satisfying the required properties according to the definition of concurrently non-malleable coin-tossing. For presentation convenience, besides the simulated key generation phase of S we describe the simulation procedure of $S^A(S_L, S_R)$ in two parts: the left part, in which S plays the role of honest left players and concurrently interacts with A; the right part, in which S plays the role of the honest right players and concurrently interacts with A. The algorithm $S^A(S_L, S_R)$ is depicted in Figure 7 (page 44).

Generation of Simulated Public-Key File F.
For $i := 1$ to $s(n)$ do:
$(TCPK_0^{(i)}, TCSK_0^{(i)}) \xleftarrow{R} TCGen(1^n)$.
$(TCPK_1^{(i)}, TCSK_1^{(i)}) \xleftarrow{R} TCGen(1^n)$.
$(TCPK_2^{(i)}, TCSK_2^{(i)}) \xleftarrow{R} TCGen(1^n)$.
$b \xleftarrow{R} \{0,1\}$.
Publish $(TCPK_0^{(i)}, TCPK_1^{(i)}, TCPK_2^{(i)})$ as the i-th public-key in the simulated public-key file F and keeps $(TCSK_b^{(i)}, TCSK_2^{(i)})$ as the corresponding secret-key while discards $TCSK_{1-b}^{(i)}$.

Straight-Line Left Simulation	**Rewinding-Ordered Right Simulation**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between S and A in the left part of CMIM with respect to a public-key $(TCPK_0^{(j)}, TCPK_1^{(j)}, TCPK_2^{(j)})$, $1 \le i, j \le s(n)$, S acts in the following way: In Stage-1, S uniformly selects $u \xleftarrow{R} \{0,1\}^n$ and sends $\alpha = TCCom(TCPK_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, S sends $S_L^{(i)}$ to A in Stage-3 (rather than sending back $u \oplus \tilde{r}_r^{(i)}$ as the honest left player does). Then, in Stage-4, using the practical concurrent zero-knowledge argument of knowledge (CZKAOK) protocol S proves that "α is the commitment to $S_L^{(i)} \oplus \tilde{r}_r^{(i)}$". The witness used by S is generated from from $TCPK_2^{(j)}$ (specifically, by running TCFake$(TCPK_2^{(j)}, TCSK_2^{(j)}, \alpha, S_L^{(i)} \oplus \tilde{r}_r^{(i)})$).	$i := 1$ Running A until receiving the first Stage-1 message, denoted α_i, marks the session of α_i the i-th session. **Label:** Suppose the i-th session is with respect to a public-key $(TCPK_0^{(j)}, TCPK_1^{(j)}, TCPK_2^{(j)})$. While "$A$ does not stop and S does not abort" do • Uniformly selects $v \xleftarrow{R} \{0,1\}^n$ and sends v to A as the Stage-2 message of the i-th session. • Runs A further and acts accordingly in any session other than the i-th session. When running into the CZKAOK phase (Stage 3-4) of the i-th session, denoted by the Stage-3 message (from A) of the i-th session $\tilde{r}_l^{(i)}$, S uses the knowledge-extractor of CZKAOK protocol to extract the witness used by A in the CZKAOK phase of the i-th session. This is achieved by rewinding A and such rewindings are called *knowledge rewindings*. If the extracted value is (t_i, s_i) such that $\alpha_i = \text{TCCom}(TCPK_2^{(j)}, t_i, s_i)$, where t_i is of length n, then S does the following: 1. rewinds A to the point A just sent α_i and sends back $S_R^{(i)} \oplus t_i$ to A. Such a rewinding is called the i-th *major rewinding*. 2. Runs A further (from the i-th major rewinding point) until receiving a new Stage-1 message from A; sets $i := i+1$ and marks the new Stage-1 message α_i and the current session of α_i the i-th session. 3. Goto **Label**
We stress that at any point of simulation, if A does not act accordingly or fails to provide a valid proof then S aborts and outputs the public-key file F and the transcript up to now.	

Figure 7. The simulation of $S^A(S_L, S_R)$

First, we show that the simulator $S^A(S_L, S_R)$ runs in expected polynomial time. Note that whenever S rewinds A in the right simulation, the interactions between S and A in the left simulation are also rewound. But since the left simulation is straight-line simulatable and all the major rewindings in the right simulation are ordered , no simultaneous rewindings take place. Since A works in polynomial-time, there are also polynomially many major rewindings in the right simulation. And during each major rewinding S also runs in expected polynomial time by running the knowledge extractor of CZKAOK. We conclude that S works in expected polynomial time in total.

Below, we show that the simulation of S (depicted in Figure 7) does satisfy the required properties of concurrently non-malleable coin-tossing: simulatability, strategy-restricted and predefinable randomness.

(1) Simulatability.

According to the definition of concurrently non-malleable coin-tossing, this needs to show that the output of $S^A(S_L, S_R)$ is computationally indistinguishable from the view of A in real concurrent executions of the protocol $\Pi =< L, R >$ (depicted in Figure 6) in the CMIM setting of BPK model. Actually, as we shall see, the output of S is identical to the view of A in real concurrent executions of $\Pi =< L, R >$ in the CMIM setting of BPK model.

We consider a hybrid experiment $H^A(S_L, S_R)$ in which a PPT algorithm H works just as S does but with the following exception that in the i-th session of the straight-line left simulation, H sends $u \oplus \tilde{r}_r^{(i)}$ to A (rather than sending back $S_L^{(i)}$ as S does) in Stage-3 and uses the corresponding randomness (used to commit to u in State-1) as the witness in Stage-4.

It is clear that the view of A in $H^A(S_L, S_R)$ is identical to the view of A in real concurrent executions of the protocol $\Pi =< L, R >$ in the CMIM setting of BPK model. It is also easy to see that the view of A in $H^A(S_L, S_R)$ is identical to the view of A in $S^A(S_L, S_R)$. Specifically, in any session of straight-line left simulation, S and H all send a random string of length n in Stage-3 and use a random decommitment string as the witness in Stage-4.

(2) Strategy-restricted and predefinable andomness.

Denote by $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots, R_L^{p(n)}\}$ the set of outputs recorded in the output S for all concurrent sessions of the left part of CMIM in BPK model and denote by $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots, R_R^{p(n)}\}$ the set of outputs recorded in the output of S for all concurrent sessions of the right part of CMIM in BPK model.

It is clear that for any i, $1 \leq i \leq p(n)$, $R_L^{(i)} = S_L^{(i)}$ that is a random string of length n. For any string $R_R^{(i)} \in R_R$, there are three cases:

Case 1. $R_R^{(i)} = S_R^{(i)}$ that is a random string of length n.

Case 2. $R_R^{(i)} \in R_L$

Case 3. Other cases.

We first show the probability that Case 3 occurs is negligible. Suppose Case 3 occurs with non-negligible probability, then with non-negligible probability there exists an i such that the output of the i-th session in the right simulation of S is neither Case 1 nor Case 2. Since the view of A in $S^A(S_L, S_R)$ is identical to the view of A in $H^A(S_L, S_R)$, then with the same probability there exists an i such that the output of the i-th session in the right simulation of H is also neither Case 1 nor Case 2. Let $F' = \{(TCPK^{(1)}, TCPK_2^{(1)}), (TCPK^{(2)}, TCPK_2^{(2)}), \cdots, (TCPK^{(s(n))}, TCPK_2^{(s(n))}), \}$, then we will construct a probabilistic algorithm E that takes F' and (S_L, S_R) as inputs and works in expected polynomial-time (with oracle access to A) to break the discrete logarithm hardness assumption by oracle

accessing to A. Actually, as we shall see, $E^A(F', S_L, S_R)$ works just as $H^A(S_L, S_R)$ does but with a modified simulated public-file generation. Specifically, in $E^A(F', S_L, S_R)$ the simulated public-file is generated from F' rather than generated by itself from scratch. For clarity, the full description of the algorithm $E^A(F', S_L, S_R)$ is depicted in Figure 8 (page 47).

It is clear that $E^A(F', S_L, S_R)$ is identical to $H^A(S_L, S_R)$. This means that with the same non-negligible probability there also exists an i such that the output of the i-th session in the right simulation of E is also neither Case 1 nor Case 2. However, by knowledge-rewinding A using the knowledge extractor of CZKAOK after the major-rewinding point of the i-th session, S will extract another different decommitment to the same Stage-1 message α_i of the i-th session. Note that from two different decommitments to the same α_i (one is extracted before the major-rewinding point of the i-th session and another is extracted after the major-rewinding point of the i-th session) E will easily computes out $TCSK_2$, which contradicts the discrete logarithm hardness assumption.

Now, all left is to show that with overwhelming probability each string in R_L can appear in R_R at most once. It is sufficient to show that with overwhelming probability R_R consists of distinct values. Suppose with non-negligible probability there exists an i such that $R_R^{(i)} = R_R^{(j)}$ for $j < i$, then by knowledge-rewinding A after the major-rewinding point of the i-th session, S will extract another decommitment $R_R^{(j)} \oplus S_R^{(i)} \oplus t_i$ to the same Stage-1 message α_i of the i-th session, where t_i is the decommitment to α_i extracted before the major-rewinding point of the i-th session. Since $S_R^{(i)}$ is a truly random value, with overwhelming probability $R_R^{(j)} \oplus S_R^{(i)} \oplus t_i \neq t_i$. Again, from two different decommitments to the same α_i E can easily extract the secret-key $TCSK_2$, which violates the hardness assumption of discrete logarithm.

Black-box concurrent player security.

We now proceed to prove the black-box concurrent player security of the protocol $\Pi =< L, R >$ depicted in Figure 6.

(1) Black-box concurrent right-player security.

Recall that in the concurrent right-player security setting, a adversary plays the role of left-players and concurrently interacts with polynomially many instances of the rith-player. But since left-players are not involved in the key-generation stage of the BPK model, the adversary playing the role of left-players keeps no secret information. It means that for any adversary A in the concurrent right-player security setting of the BPK model there exists another adversary B in the CMIM setting of the BPK model, such that what the adversary A can do in its concurrent attacks against honest right-players can also be done by B in the right part of CMIM in the BPK model. Furthermore, B can do that independently of the concurrent executions in the left part of CMIM. This means that black-box concurrent channel security in the BPK model actually implies black-box concurrent right-player security in the BPK model. Note that we have proved that the protocol $\Pi =< L, R >$ does satisfy black-box concurrent channel security in the BPK model, so it also satisfies black-box concurrent right-player security in the BPK model.

(2) Black-box concurrent left-player security.

For black-box concurrent left-player security in the BPK model, we first observe that black-box concurrent channel security in the BPK model does not imply black-box concurrent left-player security in the BPK model any longer. The reason is that in the concurrent left-player security setting of the BPK model , an adversary A firstly generates and publishes a public-key file and then concurrently interacts with polynomially many instances of the honest left-player with respect to the published public-file. Thus, the adversary keeps secret information (the corresponding secret-keys) and his behavior may also

Generation of Simulated Public-Key File F From F'.
For $i := 1$ to $s(n)$ do:
$(TCPK', TCSK') \xleftarrow{R} TCGen(1^n)$.
$b \xleftarrow{R} \{0,1\}$.
Set $TCPK_b^{(i)}$ be $TCPK'$, $TCSK_b^{(i)}$ be $TCSK'$ and $TCPK_{1-b}^{(i)}$ be $TCPK^{(i)}$.
Publish $(TCPK_0^{(i)}, TCPK_1^{(i)}, TCPK_2^{(i)})$ as the i-th public-key in the simulated public-key file F and keeps $TCSK_b^{(i)} = TCSK'$ in secret as the corresponding secret-key.

Straight-Line Left Simulation	**Rewinding-Ordered Right Simulation**
In the i-th concurrent session (ordered by the time-step in which the first round of each session is played) between E and A in the left part of CMIM with respect to a public-key $(TCPK_0^{(j)}, TCPK_1^{(j)}, TCPK_2^{(j)})$, $1 \le i, j \le s(n)$, E acts in the following way: In Stage-1, E uniformly selects $u \xleftarrow{R} \{0,1\}^n$ and sends $\alpha = TCCom(TCPK_2^{(j)}, u)$ to A. After receiving Stage-2 message, denoted $\tilde{r}_r^{(i)}$, from A, E sends $u \oplus \tilde{r}_r^{(i)}$ to A in Stage-3 just as the honest left player does. Then, in Stage-4, using the practical concurrent zero-knowledge argument of knowledge (CZKAOK) protocol E proves that "α is the commitment to u". The witness used by E is the corresponding randomness used to commit to u in State-1.	$i := 1$ Running A until receiving the first Stage-1 message, denoted α_i, marks the session of α_i the i-th session. **Label:** Suppose the i-th session is with respect to a public-key $(TCPK_0^{(j)}, TCPK_1^{(j)}, TCPK_2^{(j)})$. While "$A$ does not stop and S does not abort" do • Uniformly selects $v \xleftarrow{R} \{0,1\}^n$ and sends v to A as the Stage-2 message of the i-th session. • Runs A further and acts accordingly in any session other than the i-th session. When running into the CZKAOK phase (Stage 3-4) of the i-th session, denoted by the Stage-3 message (from A) of the i-th session $\tilde{r}_l^{(i)}$, E uses the knowledge-extractor of CZKAOK protocol to extract the witness used by A in the CZKAOK phase of the i-th session. This is achieved by rewinding A and such rewindings are called *knowledge rewindings*. If the extracted value is (t_i, s_i) such that $\alpha_i = \text{TCCom}(TCPK_2^{(j)}, t_i, s_i)$, where t_i is of length n, then E does the following: 1. rewinds A to the point A just sent α_i and sends back $S_R^{(i)} \oplus t_i$ to A. Such a rewinding is called the i-th *major rewinding*. 2. Runs A further (from the i-th major rewinding point) until receiving a new Stage-1 message from A; sets $i := i+1$ and marks the new Stage-1 message α_i and the current session of α_i the i-th session. 3. Goto **Label**
We stress that at any point of simulation, if A does not act accordingly or fails to provide a valid proof then E aborts and outputs the public-key file F and the transcript up to now.	

Figure 8. The algorithm $E^A(F', S_L, S_R)$

depend on such secret information. An algorithm without the knowledge of the secret information (like adversaries in the CMIM setting of the BPK model) may not do what the adversary A can do.

According to the definition of black-box concurrent left-player security, we need to construct a probabilistic (expected) polynomial-time simulator S such that for any PPT $s(n)$-concurrent adversary A that plays the role of ritht-player and concurrently interacts with many instances of the honest left-player in at most $s(n)$ sessions, the output of $S^A(R_S)$ satisfies the properties of simulatability and predefinable randomness, where $R_S = \{r_S^{(1)}, r_S^{(2)}, \cdots, r_S^{(s(n))}\}$ be a set of $s(n)$ random strings of length n each. Without loss of generality, we also suppose A generates and publishes a public-file that contains $s(n)$ public-keys of the form $(TCPK_0^{(i)}, TCPK_1^{(i)}, TCPK_2^{(i)})$, $1 \leq i \leq s(n)$. $S^A(R_S)$ works in at most $s(n)+1$ phases. We remark that in the i-th session of each phase, $1 \leq i \leq s(n)$, S sends to A $r_S^{(i)}$ rather than $r_l \oplus r_r$ in Stage-3, where r_l is the value committed in Stage-1 of the i-th session by S and r_r is the value sent by A to S in Stage-2. In each phase, S either successfully gets a simulated transcript or "breaks" a new public-key in the sense that S can extract the corresponding secret-key $TCSK_b$ (according to the special soundness of Σ-protocol, this is achieved by rewinding A to get two different accepting conversations w. r. t. the same first message sent by A in Stage-4). Once a public-key $(TCPK_0, TCPK_1, TCPK_2)$ is broken (that is, S learns $TCSK_b$), then S uses the witness generated from $TCSK_b$ in Stage-4.

Since S runs in at most $s(n)+1$ phases, and during each phase S also works in expected polynomial time, it is easy to see that S runs in expected polynomial time in toto.

Due to the concurrent perfect zero-knowledge property of Stage-4, it is clear that output of $S^A(R_S)$ is identical to the view of V^* in its real interactions. Denote by $R_A = \{r_A^{(1)}, r_A^{(2)}, \cdots, r_A^{(s(n))}\}$ the outputs (recorded in the output of $S^A(R_S)$) of all the concurrent sessions between S and A, then it is also easy to see that $R_A = R_S$.

□

7 A Survey of Subsequent Discoveries

After the original publication in January 2004, two and a half years has passed. In this subsection, we make a survey on subsequent discoveries that closely related to this work.

Firstly, the two CZK-CS protocols are found flawed independently in [29, 70]. [29] fixed the flaw very nicely, following the protocol structure of the first protocol. The first protocol is also extended to transform any public-coin HVZK into generic yet practical (statistical) ZK arguments in the standard model *with optimal communication and computational complexity incurred*. [74]. Very recently, [26] fixed the flaw of the second protocol with the same sprit of [29]. Unfortunately, in [70], we show that the fixed protocol of [29] does not provide concurrent verifier security, though it is concurrently sound. The reason is that although a malicious prover cannot convince of a false statement, but it can convince a true statement but without any knowledge of the witness! In next section, we show the fixed protocol of [26] suffers the same problem, showing that it does not provide concurrent verifier security. In [70], we provide RZK with full concurrent verifier security under subexponential hardness assumptions. In particular, we get OWF-based, round-optimal but general hardness based, and highly practical solutions. This leaves a main problem whether we can achieve full concurrent verifier security for concurrent/resettable ZK under standard hardness assumptions in the BPK model. Note that this problem is very tricky and subtle, witnessed by the evolution history in the past two and a half years. Moreover, it even turns out to be tricky to provide a formal definition of full concurrent verifier security in BPK model *under standard assumption* (note that formal definition under subexponential assumption is given in [70]).

In this work, we solve this main problem.

8 Some Observations on the Protocol of [26]

We briefly recall the fixing of [26]: In Stage-2 of the protocol of Figure 5, the prover commits to a string, denoted $C = Com(e)$, then prover proves to verifier that either $x \in L$ or C commits to one of verifier's secret-keys.

Then, we show that the protocol of [26] does not achieve full concurrent verifier security. Specifically, we demonstrate attacks that enable a malicious P^* to convince V of a true statement but without knowing any witness of the proved statement.

The attacks are very similar to those developed in [70, 71] against the fixed protocol of [29].

The fixing is in the same sprit of [29] that fixes the fist protocol, and provides concurrent soundness.

But, using the same attack strategies presented in [70, 71], one can easily convince a true statement for the language: "$\{((PK_0, PK_1), w)\}$" such that either w is SK_0 or SK_1. Specifically, a malicious prover make a simulated proof for "C commits to one of secret-keys", and then by interleaving two concurrent sessions, it can malleate the proof given by V on (PK_0, PK_1) in one session into a successful Stage-2 interactions in another concurrent session. Note that P^* does not know any of verifier's secret keys. The second concurrent interleaving and malleating attacks of [70] also can be trivially applied to the fixed protocol of [26]. Details can be found in [70, 71].

These observations showing that the fixed protocols of [29, 26] are still *not* secure in concurrent settings.

9 Full Concurrent Verifier Security for Concurrent/Resettable ZK under Standard Assumptions in the BPK Model

Then, how to get full concurrent verifier security for CZK/rZK under standard assumptions in the BPK model? Or, in other words, how to modify the flawed protocols of the original version of this paper into *correct* and *right* ones?

We solve this main open problem in this section. But, before presenting our protocols, we need to first give a proper definition of full concurrent verifier security. Note that the attacks developed in [70, 71] are of nature of man-in-the-middle attacks. That is, they are related to non-malleability of protocols.

The informal intuition for full concurrent verifier security is: for any x if P^* can convince V (with public-key PK) of "$x \in L$", then it must know a witness for this fact. More formally, for any x if P^* can convince V (with public-key PK) of "$x \in L$", then there exists a PPT knowledge-extractor that outputs a witness for $x \in L$. But, such a definition does not fully work in the public-key model. For example, the language may be related to PK, and thus the extracted witness may be related to SK. But, in knowledge-extraction the PPT extractor may have already possessed SK. To solve this subtle problem, we require the extracted witness is *independent* of SK. But, the difficulty here is: How to formalize such independence, in particular, when SK is *unique*. The solution is we consider the message space (distribution) of SK, and such independence is defined as follows: for any polynomial-time computable relation R, let SK be the real secret-key and SK' is an element randomly sampled from the message space of secret-keys, then we require that the probability $\Pr[R(w, SK) = 1]$ is negligibly close to $\Pr[R(w, SK') = 1]$, where w is the witness extracted by the PPT knowledge extractor.

Remark: If V has a pair of independent keys (PK_0, PK_1) as in our case (when they are formed with pseudorandom generators such that PK does not reveal the secret-key), the independence can also be formalized as $\Pr[R(w, SK_0) = 1]$ is negligibly close to $\Pr[R(w, SK_1 = 1]$. This in particular guarantees that P^*'s interactions cannot depends on the secret-key used by V, and thus cannot malleate V's proofs.

Definition 9.1 (concurrent knowledge-extraction (CKE) in the public-key model) *We say that a protocol $\langle P, V\rangle$ for an $\mathcal{NP}$-language L (with $\mathcal{NP}$-relation R_L) is* concurrently knowledge-extractable *in the BPK model, if there exists a probabilistic polynomial-time knowledge-extractor E such that for any sufficiently large n, any binary relation R computable in time $poly(n)$, any PPT honest verifier V with registered public-key PK and corresponding secret-key SK, and any (whether false or true) string x of length n, for all positive polynomials s and all s-concurrent malicious prover P^* (as defined in Section ?), if P^* can convince the honest verifier of the statement "$x \in L$" with non-negligible probabilities in the concurrent interactions, then the following hold:*

- *With non-negligible probabilities, $(x, E(1^n, PK, SK)) \in R_L$. That is, The knowledge-extractor, on input $(1^n, PK, SK)$, will output a witness w for $x \in L$ also with non-negligible probabilities.*
- *Furthermore, $\Pr[R(E(1^n, PK, SK), SK) = 1]$ is negligibly close to $\Pr[R(E(1^n, PK, SK), SK') = 1]$, where SK' is randomly and independently selected from the space of SK. Specifically, for any values v_1 and v_2 from the space of secret-keys, $Prob[SK = v_1 \ \wedge \ SK' = v_2] = Prob[SK = v_1] \cdot Prob[SK' = v_2]$. This captures that the extracted witness is "independent" of the secret-key used by E. Note that we do not require that SK' is independent of the public-key PK.*

The probability is taken over the the randomness of V in the key-generation stage (i.e., the randomness for generating (PK, SK)) and in all the $s(n)$ verification stages, the randomness of E and the choice of SK'. Here we assume, without loss of generality, that P^* is deterministic. The algorithm E is called a black-box knowledge-extractor if it only oracle accesses to P^*, and a non-black-box knowledge-extractor if it takes the codes of P^* as its input.

Now, we return back for achieving full concurrent verifier security for CZK under standard assumptions in BPK model.

Protocol-1: In the fixed protocol of [29], we require public-keys to be generated by PRG [6]. Specifically, we require $PK_b = PRG(r_b)$, where $b \in \{0, 1\}$ and r_b is a random string. Also, we require that prover does not directly proves $x \in L$ with Σ-protocol, where random challenges are set by a coin-tossing mechnism. Rather, it first commits to a witness using a perfectly-binding commitment scheme, i.e., $C_w = Com(w)$, then it proves to V that committed value is a witness for $x \in L$.

Protocol-2: In the fixed protocol of [26], we require public-keys to be generated by PRG. Specifically, we require $PK_b = PRG(r_b)$, where $b \in \{0, 1\}$ and r_b is a random string. Also, we require that in Stage-2, P first computes $C_w = Com(w)$ using a perfectly-binding commitment scheme, and then proves to V that either the committed value in C_w is a witness for $x \in L$ OR another commitment C commits to a secret-key of V's secret-keys.

We describe the Protocol-2 and the proof in details. The protocol is depicted in Figure-9.

We examine the underlying complexity assumptions and round-complexity of the protocol depicted in Figure-9. PRG can be based on any one-way function (OWF) [**?**][7]. Perfectly-binding commitments can be based on any one-way permuation (OWP) with one-round commitment stage [44], or based on any OWF but with a 2-round commitment stage in which the commitment receiver sends public-coins in the first round [**?**]. Public-coin partial witness independent WIPOK protocols for $\mathcal{NP}$ can be based on any OWP with three rounds, or based on any OWF but with four rounds [10, **?**]. Note that the protocol depicted in Figure-9 can be easily combined into four rounds (that is optimal) if the partial witness independent public-coin WIPOK in Stage-1 is of three rounds, or into five rounds if Stage-1 is

[6] PRG can be replaced by any OWF

[7] Actually, PRG can be just replaced by any OWF

The protocol $\langle P, V \rangle$

Key Generation. Let $PRG : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator, where n is the security parameter. Each verifier selects two random strings s_0, s_1 from $\{0,1\}^n$, computes $y_0 = PRG(s_0)$ and $y_1 = PRG(s_1)$, publishes $PK = (y_0, y_1)$ as its public-key. For its secret-key, the verifier randomly selects a bit $b \leftarrow \{0,1\}$ and keeps $SK = s_b$ in secret as its secret-key while discarding s_{1-b}. Note that $PRG(s)$ is actually a OWF [44].

Common input. An element $x \in L \cap \{0,1\}^n$. Denote by R_L the corresponding $\mathcal{NP}$-relation for L.

P private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^n$ for $x \in L$. Here, we assume w.l.o.g. that the witness for any $x \in L \cap \{0,1\}^n$ is of the same length n (in particular, consider the $\mathcal{NP}$-complete language DHC).

Stage-1. V proves to P that it knows a preimage to one of y_0, y_1, by running a *partial witness independent* public-coin WI proof of knowledge (WIPOK) protocol for $\mathcal{NP}$ in which V plays the role of knowledge prover. The witness used by V in this stage is s_b.

Stage-2. If V successfully finishes Stage-1, P does the following: it computes $c_w = C(w, r_w)$ and $c_{sk} = C(0^n, r_{sk})$, where C is a perfectly-binding commitment scheme and r_w and r_{sk} are the randomness used for commitments.

Stage 3. Define a new $\mathcal{NP}$-language $L' = \{(x, y_0, y_1, c_w, c_{sk}) | (\exists (w, r_w) \ s.t. \ c_w = C(w, r_w) \wedge (x, w) \in R_L) \vee (\exists (w, r_{sk}, b) \ s.t. \ c_{sk} = C(w, r_{sk}) \wedge y_b = PRG(w) \wedge b \in \{0,1\})\}$. Then, P proves to V that it knows a witness for $(x, y_0, y_1, c_w, c_{sk}) \in L'$, by running a public-coin WI proof of knowledge (WIPOK) protocol for $\mathcal{NP}$. The witness used by P is (w, r_w).

Figure 9. The cZK-CKE argument for $\mathcal{NP}$ in the BPK model.

of four rounds. Thus, we conclude that the protocol described in Figure-9 can be based on any OWP with optimal round-complexity, or based on any OWF but with five rounds.

On the necessity of c_w and c_{sk}. We stress that the mandating the perfectly-binding commitments c_w and c_{sk} play a crucial role for achieving concurrent knowledge-extraction in the public-key model. In particular, protocol variants without any of them will not provide concurrent verifier security in the public-key model. Specifically, for protocol variants without mandating any of c_w and c_{sk}, a slight modification of the concurrent interleaving and malleating attack can enable a malicious prover convince the honest verifier of a (false or true) statement but without any witness for the statement being proved.

Theorem 9.1 *Assuming any OWF (resp., any OWP), there is a 5-round (resp., 4-round) concurrently knowledge-extractable concurrent ZK argument for $\mathcal{NP}$ in the BPK model.*

Proof (sketch). The completeness of the protocol $\langle P, V \rangle$ can be easily checked. Below, we focus on the proof of concurrent zero-knowledge and concurrent knowledge-extraction.

Concurrent zero-knowledge.

We first consider a mental simulator M that takes all secret-keys corresponding to all public-keys registered in the public-key file, in case the corresponding secret-keys exist.

For any $s(n)$-concurrent malicious verifier V^*, M runs V^* as a subroutine on input $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ (where x_i might equal x_j, $1 \leq i, j \leq s(n)$ and $i \neq j$), the public file $F = \{PK_1, \cdots, PK_{s(n)}\}$ and all assumed existing secret-keys. M works just as the honest prover in Phase-1 of any session. In Phase-2 of any session on a common input x_i and with respect to a public-key PK_j (i.e, the i-th session w.r.t PK_j, $1 \leq i, j \leq s(n)$), M computes $c_w^{(i)} = C(0^n, r_w^{(i)})$ and $c_{sk}^{(i)} = C(SK_j, r_{sk}^{(i)})$, where SK_j is the secret-key corresponding to PK_j for which we assume it exists and M knows. Then, M runs the WIPOK protocol with V^* in Phase-3 of the session with $(SK_j, r_{sk}^{(i)})$ as its witness.

To show the output of M is indistinguishable from the view of V^* in real concurrent interactions, we consider another mental simulator M'. M' takes both the witnesses for $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ and all the secret-keys corresponding to public-keys registered in F (in case the corresponding secret-keys exist). M' works just as M does, but with the following exception: in Phase-2 of the i-th session on common input x_i w.r.t PK_j, $1 \leq i, j \leq s(n)$, M' computes $c_w = C(w_i, r_w^{(i)})$, where w_i is the witness for the common input x_i. Note that the witness used by M' in Phase-3 is still SK_j, just as M does. The output of M' is indistinguishable from that of M is from the computational hiding property of the perfectly-binding commitment scheme C used in Phase-2.

We now consider another mental simulator M'' that mimics M' with the following exception: in Phase-3 of the i-th session on common input x_i w.r.t PK_j, $1 \leq i, j \leq s(n)$, the witness used by M'' is w_i, rather than SK_j as used by M'. The output of M'' is indistinguishable from that of M' by the WI property of underlying WIPOK protocol of Phase-3. Also, the output of M'' is also indistinguishable from the view of V^* in real concurrent interactions by the computational hiding property of the underlying perfectly-binding commitment scheme C used in Phase-2.

This establishes that the output of M is indistinguishable from the view of V^* in real concurrent interactions. To build a PPT simulator S from scratch, where S does not know any secret-key corresponding to public-keys in the public file, we resort to the technique developed in [13]. Specifically, S works in $s(n)+1$ phases. In each phase, S either successfully finishes the simulation or "covers" a new public-key, in case V^* successfully finishes the Phase-1 interactions w.r.t. that public-key, for which it has not known the corresponding secret-key by now. Key covering is guaranteed by the POK property of Phase-1 interactions. More details are referred to [13].

Concurrent knowledge-extraction.

For any sufficiently large n, any $x \in \{0,1\}^n$, any positive polynomial $s(n)$ and any $s(n)$-concurrent malicious prover P^*, suppose with non-negligible probability $p(n)$ there exists a j, $1 \leq j \leq s(n)$, such that P^* can convince the honest verifier V (with public-key $PK = (y_0, y_1)$ and secret-key $SK = s_b$) of the statement "$x \in L$" in the j-th session, we construct a polynomial-time knowledge-extractor E as follows. Note that s_b is randomly chosen from $\{0,1\}^n$ and b is randomly chosen from $\{0,1\}$.

E takes (PK, SK) as input, and randomly selects i from $\{1, \cdots, s(n)\}$ (that is, E randomly guesses the session j). E runs P^* as a subroutine, and works just as the honest verifier does. For the i-th session initiated by P^* on a common input x_i, in case P^* finishes Phase-3 interactions of the i-th session, we denote by $a_P^{(i)}, e_P^{(i)}, z_P^{(i)}$, the first-round, the second-round and the third-round messages of Phase-3 of the i-th session, and by $c_w^{(i)}, c_{sk}^{(i)}$, the commitments sent by P^* in Phase-2 of the i-th session. Then, E rewinds P^* to the point that P^* just sent $a_P^{(i)}$, sends P^* a new random challenge $\hat{e}_P^{(i)}$, and runs P^* further. If E receives another final message $\hat{z}_P^{(i)}$ of the i-th session again, then from $(a_P^{(i)}, e_P^{(i)}, z_P^{(i)})$ and $(a_P^{(i)}, \hat{e}_P^{(i)}, \hat{z}_P^{(i)})$ E will get: either $(w, r_w^{(i)})$ such that $c_w^{(i)} = C(w, r_w^{(i)})$ and $(x_i, w) \in R_L$ OR $(w, r_{sk}^{(i)}, \sigma)$ such that $c_{sk}^{(i)} = C(w, r_{sk}^{(i)})$ and $y_\sigma = PRG(w)$ and $\sigma \in \{0,1\}$. This is guaranteed by the special soundness of the WIPOK protocol of Phase-3 (e.g., when it is implemented by Blum's WIPOK protocol for DHC). In this case, E outputs w. In any other case, E outputs a failure symbol $\perp$.

It is easy to see with probability $\frac{1}{s(n)}$ E successfully guesses the session j (i.e., $i = j$). Note that

in this case $x_i = x$. Conditioned on E successfully guesses j, with probability about $p(n)^2$ (except for the exponentially small probability that $\hat{e}_P^{(i)} = e_P^{(i)}$), E will extract a witness w by the POK property of Phase-3 and by the assumption that with probability $p(n)$ P^* can convince V (equivalently, E) of $x \in L$ in the j-th session. Thus, with non-negligible probability $\frac{p(n)^2}{s(n)}$ E outputs w, and conditioned on E outputs w there are three cases:

Case-1. $\sigma = 1 - b$ and $y_\sigma = PRG(w)$.

Case-2. $\sigma = b$ and $y_\sigma = PRG(w)$.

Case-3. $(x, w) \in R_L$.

Case-1 can occur only with negligible probabilities, due to the one-wayness of y_{1-b}. The following lemma shows Case-2 occurs also with negligible probabilities.

Lemma 9.1 *Case-2 occurs with negligible probabilities.*

Proof (of Lemma 9.1). We consider two experiments: E_0 and E_1. For each $b \in \{0, 1\}$, E_b mimics the behaviors of $E(PK, SK)$, where $PK = (y_0, y_1)$ and $SK = s_b$ such that $y_b = PRG(s_b)$. Suppose Case-2 occurs with non-negligible probabilities, then there must exists a bit b such that E_b will output a preimage of y_b with non-negligible probabilities (otherwise, Case-2 will trivially occur with negligible probabilities). Without loss of generality, we assume $b = 0$. That is, E_0 outputs the preimage of y_0 with non-negligible probability (and outputs the preimage of y_1 with negligible probabilities). Now we consider the output of of E_1: first, it outputs the preimage of y_0 also with negligible probability; thus, with non-negligible probabilities E_1 outputs either the preimage of y_1 or the witness for $x \in L$. Note that here we cannot directly conclude that E_1 will certainly output the preimage of y_1 with non-negligible probabilities, as we cannot rely on the assumption that $x \notin L$. Then, we show how to contradict the WI or partial witness independence properties of the POK protocol of Phase-1.

We define a series of hybrid mental experiments $H_1, \cdots, H_{s(n)}$ as follows: for any k, $1 \leq k \leq s(n)$, H_k takes $(PK = (y_0, y_1), s_0, s_1)$ as input, where $y_b = PRG(s_b)$ for $b \in \{0, 1\}$. That is, H_k takes both the two secret-keys as input. H_k mimics the behaviors of E but with the following exceptions: In Phase-1 of the first k sessions H_k uses s_1 as its witness; and in Phase-1 of the rest $s(n) - k$ sessions it uses s_0 as the witness. Note that H_0 equals the experiment E_0, and $H_{s(n)}$ equals the experiment E_1. As we assume that $H_0 (= E_0)$ will output the preimage of y_0 with non-negligible probability (but output the preimage of y_1 with negligible probability), and that $H_{s(n)} (= E_1)$ will output either a preimage of y_1 or a witness for $x \in L$ with non-negligible probability (but output the preimage of y_0 only with negligible probability). By hybrid arguments, we conclude that there must exists a k, $1 \leq k \leq s(n)$, such that H_{k-1} outputs the preimage of y_0 with non-negligible probability and H_k outputs the preimage of y_0 with negligible probability. Then we show how to break the WI property or the partial witness-independent property of the POK protocol of Phase-1, by considering another experiment B.

The input of B is $((y_0, y_1), s_0, s_1, k)$. B runs P^* as a subroutine internally and interacts externally with a knowledge-prover $\hat{P}_k$ by running the underlying partial witness-independent public-coin WIPOK protocol on (y_0, y_1) used in Phase-1, in which B plays the role of knowledge verifier. The knowledge prover $\hat{P}_k$ takes a secret input s_b for a random bit b such that $y_b = PRG(s_b)$. B mimics the experiment of H_k with the following exceptions (note that H_k in turn mimics E that rewinds P^* in the i-th session): The Phase-1 interactions of the k-session are emulated externally by forwarding P^*'s messages to the external knowledge-prover $\hat{P}_i$, and returns responses from $\hat{P}_i$ to P^*.

Then according to above hybrid arguments, if $\hat{P}_k$ uses s_1 as its witness, then the experiment B is identical to H_k, and thus B outputs the preimage of y_0 with negligible probability; On the other

hand, if $\hat{P}_k$ uses s_0 as its witness, then B is identical to H_{k-1} and will output the preimage of y_0 with non-negligible probability. Now, we consider two cases:

Case 2.1. The external interactions with $\hat{P}_i$ has finished before the rewinding point in the i-th session.

Case 2.2. The external interactions with $\hat{P}_i$ has not finished before the rewinding point in the i-th session. This means that $\hat{P}_i$ sends *at most* the first-round message (it may be the case that $\hat{P}_i$ is initiated after the rewinding point). Note that the concurrent interleaving and malleating attack described in Section ? is just of this case.

If Case 2.1 occurs, then we break the WI property of the partial witness-independent WIPOK protocol as follows: If B outputs the preimage of y_0, then we also output 0; in any other cases, we output a random bit δ. According to above hybrid arguments, if $\hat{P}_k$ uses s_0 as its witness, then we will output 0 with probability that is non-negligibly bigger than 1/2; on the other hand, if $\hat{P}_k$ uses s_1 as its witness, then we will output 0 with probability negligibly close to 1/2. As B works in polynomial-time, this contradicts the WI property of the underlying protocol.

If Case 2.2 occurs, then we know B sees *at most* the first-round message of the underlying partial witness-independent WIPOK protocol. As the first-round message is generated *independently* of the witness used by $\hat{P}_i$, and the witness used by $\hat{P}_i$ is s_b *for a random chosen bit* $b \in \{0,1\}$, we get a contradiction that no one can successfully guess the bit b.

By removing Case-1 and Case-2, we conclude now that for any x if P^* can convince of the honest verifier of the statement "$x \in L$", then the knowledge-extractor E will output a witness for $x \in L$ also with non-negligible probability. This establishes the first property of the knowledge-extractor E. To finish the proof, we need to further show that the extracted witness is independent of the secret-key used by E. As s_{1-b} is randomly and independently chosen. It is enough to establish the following lemma.

Lemma 9.2 *For any polynomial-time computable relation R, $\Pr[R(E((y_0,y_1),s_b),s_b)=1]$ is negligibly close to $\Pr[R(E((y_0,y_1),s_b),s_{1-b})=1]$, where $PK=(y_0,y_1)$ is the public-key such that $y_0=PRG(s_0)$ and $y_1=PRG(s_1)$ and $SK=s_b$ is the secret-key for a random bit b from $\{0,1\}$ and $SK'=s_{1-b}$ is randomly and independently chosen from the space of SK.*

Proof (of Lemma 9.2). $\Pr[R(E((y_0,y_1),s_b),s_b)=1]$ is negligibly close to $\Pr[R(E((y_0,y_1),s_b),s_{1-b})=1]$, where $PK=(y_0,y_1)$ is the public-key such that $y_0=PRG(s_0)$ and $y_1=PRG(s_1)$ and s_b the secret-key SK for a random bit b from $\{0,1\}$. We consider the two experiments E_0 and E_1 presented at the beginning of the proof of Lemma 9.1. We first show that $\Pr[R(E_0((y_0,y_1),s_1),s_1)=1]$ is negligibly close to $\Pr[R(E_1((y_0,y_1),s_0),s_1)=1]$. Otherwise, by using similar hybrid arguments and proof procedure as in the proof of Lemma 9.1, we can break either the WI property or the partial witness independence property of the underlying protocol used in Phase-1. Similarly, we get that $\Pr[R(E_0((y_0,y_1),s_1),s_0)=1]$ is negligibly close to $\Pr[R(E_1((y_0,y_1),s_0),s_0)=1]$. Thus, we conclude that $\Pr[R(E((y_0,y_1),s_b),s_b)=1]$ is negligibly close to $\Pr[R(E((y_0,y_1),s_{1-b}),s_b)=1]$. We stress that the perfectly-binding commitment c_w of Phase-2 plays a crucial role in the proof. By defining $\upsilon=1-b$, we get It is easy to see that $\Pr[R(E((y_0,y_1),s_{1-b}),s_b)=1]=\Pr[R(E((y_0,y_1),s_\upsilon),s_{1-\upsilon})=1]$ that is just identical to $\Pr[R(E((y_0,y_1),s_b),s_{1-b})=1]$, where both b and υ are random bits. □ □

9.1 Geting full concurrent verifier security for rZK under standard assumptions

The idea is to use the non-black-box techniques of [1, 4, 63], which is suggested in [26].

Specifically, V first commits to a random challenge on the top of the two protocols, in Stage-2, rather than decommitting directly the random challenge, V sends a random string and proves to P that it is

The protocol $\langle P, V\rangle$
Key Generation. On security parameter n, each verifier V V selects randomly strings s_0, s_1 from Z_q, computes $(y_0 = f_{p,q,g}(s_0), y_1 = f_{p,q,g}(s_1))$. Finally, V registers $PK = (y_0, y_1)$ in a public file F as its public-key, and keeps $SK = s_b$ as its secret-key for a random bit $b \in \{0,1\}$.
Common input. $x \in Z_p^*$ of order q. **P private input.** An $\mathcal{NP}$-witness $w \in Z_q$ such that $x = g^w \mod p$.
Stage-1. V proves to P that it knows a preimage to one of y_0, y_1, by running the OR-proof of the Σ-protocol of [67] for DLP (that is partial witness independent). The witness used by V in this stage is s_b. **Stage-2.** If V successfully finishes Stage-1, P does the following: it computes $c_w = C(w, r_w)$ and $c_{sk} = C(0^{\|q\|}, r_{sk})$, where C is a perfectly-binding commitment scheme to be specified below, and r_w and r_{sk} are the randomness used for commitments. **Stage-3.** Define a new $\mathcal{NP}$-language $L' = \{(x, y_0, y_1, c_w, c_{sk}) \mid (\exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge x = g^w \mod p) \vee (\exists (w, r_{sk}, b)\ s.t.\ c_{sk} = C(w, r_{sk}) \wedge y_b = f_{p,q,g}(w) \wedge b \in \{0,1\})\}$. Then, P proves to V that it knows a witness for $(x, y_0, y_1, c_w, c_{sk}) \in L'$, by running a 3-round public-coin WIPOK protocol for $\mathcal{NP}$. The witness used by P is (w, r_w).

Figure-10. The CZK-CKE argument for $\mathcal{NP}$ in the BPK model.

the value committed on the top, *using Barak's non-black-box ZK*. To get resettable ZK, all randomness used by P are got by applying a PRF on the transcript. For provable security, we need the one-many simulatability of [63, 26]. Details are postponed to the full version of this work.

10 DDH-Based Practical Implementation of CZK-CKE for DLP in the BPK Model

In the practical implementation of CZK-CKE for NP, we use the DLP OWF (indexed by (p, q, g)): $f_{p,q,g}(x) = g^x \mod p$, where p and q are primes, $p = 2q + 1$ and $|p| = n$, and g is an element of Z_p^* of order q. We also assume the DDH assumption holds on the cyclic group indexed by (p, q, g) (i.e., the sub-group of order q of Z_p^*). Now, we re-depicted the CZK-CKE protocol for DLP (that is depicted in Figure 10):

For practical implementation of the protocol depicted in Figure 10, we need a practical implementation of the perfectly-binding commitment scheme used in Stage-2, and more importantly, a practical implementation of the Σ-protocol used in Stage-3.

For practical perfectly-binding commitment scheme, we use the DDH-based ElGamal (non-interactive) commitment scheme [36]. To commit to a value $v \in Z_q$, the committer randomly selects $u, r \in Z_q$, computes $h = g^u \mod p$ and sends $(h, \bar{g} = g^r, \bar{h} = g^v h^r)$ as the commitment. The decommitment information is (r, v). Upon receiving the commitment $(h, \bar{g}, \bar{h})$, the receiver checks that $h, \bar{g}, \bar{h}$ are elements of order q in Z_p^*. It is easy to see that the commitment scheme is of perfectly-binding. The computational hiding property is from the DDH assumption on the subgroup of order q of Z_p^* (for more details, see [36]).

For the practical Σ-protocol of Stage-3, by the Σ_{OR}-technique we need the following two practical Σ-protocols:

- A practical Σ-protocol that, given $x, c_w = (h, \bar{g}, \bar{h})$, proves the knowledge of (w, r) such that $x = g^w \mod p$ and $\bar{g} = g^r \mod p$ and $\bar{h} = g^w h^r \mod p$.
- A practical Σ-protocol that, given $y_0, y_1, c_{sk} = (h, \bar{g}_{sk}, \bar{h}_{sk})$, proves the knowledge (w, r) such that **either** $y_0 = g^w \mod p$ and $\bar{g}_{sk} = g^r \mod p$ and $\bar{h}_{sk} = g^w h^r \mod p$ **or** $y_1 = g^w \mod p$ and $\bar{g}_{sk} = g^r \mod p$ and $\bar{h}_{sk} = g^w h^r \mod p$.

Again, by the Σ_{OR}-technique, if we have a practical Σ-protocol of the first type, then we can also have a practical Σ-protocol of the second type. Thus, to get a practical implementation of the protocol depicted in Figure 10, all we need now is to develop a practical Σ-protocol of the first type. Based on (but different from) the Σ-protocol for DLP [67], such Σ-protocol is described below.

Common input: $(p, q, g, x, h, \bar{g}, \bar{h})$, where $x, h, \bar{g}, \bar{h}$ are all elements of order q in Z_p^*.

Prover's private input: $w, r \in Z_q$ such that $x = g^w \mod p$ and $\bar{g} = g^r \mod p$ and $\bar{h} = g^w h^r \mod p$.

Round-1: The prover P randomly selects $t \in Z_q$, computes $a_0 = g^t \mod p$ and $a_1 = h^t \mod p$, sends (a_0, a_1) to the verifier V.

Round-2: V responds back a random challenge e taken randomly from Z_q.

Round-3: P computes $z_0 = t + we \mod p$ and $z_1 = t + re \mod p$, and sends back (z_0, z_1) to V.

Verifier's decision: V accepts if and only if: $g^{z_0} = a_0 x^e \mod p$ and $g^{z_1} = a_0 \bar{g}^e \mod p$ and $h^{z_1} = a_1 (\bar{h}/x)^e \mod p$.

Now, we proceed to analyze the properties of the above protocol:

Special soundness: From two accepting conversations w.r.t. the same Round-1 messages, $\{(a_0, a_1), e, (z_0, z_1)\}$ and $\{(a_0, a_1), e', (z_0', z_1')\}$, we can compute $w = \frac{z_0 - z_0'}{e - e'}$, and $r = \frac{z_1 - z_1'}{e - e'}$.

Special HVZK: The SHVZK simulator S works as follows: on a given random challenge $e \in Z_q$, it randomly selects z_0, z_1 from Z_q, then it sets $a_0 = g^{z_0} x^{-e}$ and $a_1 = g^{z_1} \bar{g}^{-e} = h^{z_1} (\bar{h}/x)^{-e}$.

10.1 Practical implementation based on the Micciancio-Petrank commitment scheme

In [57] Micciancio and Petrank give another implementation of DDH-based perfectly-binding commitment scheme. To commit to a value $v \in Z_q$, the committer sends $(h, \bar{g} = g^r, \bar{h} = h^{v+r})$. The decommitment information is (r, v). We then present another practical Σ-protocol for our CZK-CKE protocol using the Miccinancio-Petrank DDH-based commitment scheme.

Common input: $(p, q, g, x, h, \bar{g}, \bar{h})$, where $x, h, \bar{g}, \bar{h}$ are all elements of order q in Z_p^*.

Prover's private input: $w, r \in Z_q$ such that $x = g^w \mod p$ and $\bar{g} = g^r \mod p$ and $\bar{h} = h^{w+r} \mod p$.

Round-1: The prover P computes $H = h^w \mod p$, randomly selects $t \in Z_q$, computes $a_0 = g^t \mod p$ and $a_1 = h^t \mod p$, sends (H, a_0, a_1) to the verifier V.

Round-2: V responds back a random challenge e taken randomly from Z_q.

Round-3: P computes $z_0 = t + we \mod p$ and $z_1 = t + re \mod p$, and sends back (z_0, z_1) to V.

Verifier's decision: V accepts if and only if: $g^{z_0} = a_0 x^e \mod p$ and $g^{z_1} = a_0 \bar{g}^e \mod p$ and $h^{z_0} = a_1 H^e \mod p$ and $(h)^{z_1} = a_1(\bar{h}/H)^e \mod p$.

For the above protocol, completeness and SHVZK can be easily checked. Below, we focus on special soundness.

Special soundness: Given two accepting conversations $\{(H, a_0, a_1, a_2), e, (z_0, z_1)\}$ and $\{(H, a_0, a_1, a_2), e', (z_0', z_1')\}$, one can efficiently compute out: $w = \frac{z_0 - z_0'}{e - e'}$ and $r = \frac{z_1 - z_1'}{e - e'}$.

11 Concurrent Non-Malleability in the Public-Key Model, Revisited

In the rest of this paper, we continue the research line for achieving concurrently non-malleable *interactive* cryptographic protocols *in the public-key model* (to our knowledge, this research line is started in this work).

In this section, we first present, inspired by the concurrent interleaving attacks against the ZK protocols of [40, 29, 26] presented in [71, 70], concurrent man-in-the-middle attacks against the ZK protocol of [27]. This is somewhat surprising, as the ZK protocol of [27] is proved to be concurrently non-malleable, i.e., secure against man-in-the-middle attacks, in the public-key model. In particular, the concurrent non-malleability in [27] is clearly motivated to achieve ZK protocols that is secure in the sense that for any statement proved by an MIM adversary the adversary mush "know" a witness for the statement being proved. Our attacks show that the ZK protocol of [27] does not achieve the claimed security goal.

We then investigate the subtleties in the proof and security model and formulation in [27]. We show that the concurrent non-malleability *FOR PUBLIC-KEY MODELS* formulated in [27] (actually and in all previous works including the early version of this work) is incomplete. We clarify the formulation subtleties and then give new formulation of CNM for cryptographic protocols running in the public-key model, *particularly inspired by our CKE formulation presented in Section 9.*

11.1 Concurrent man-in-the-middle attacks on the DDL ZK protocol of [27]

Let us first recall the protocol structure of the CZK protocol of [27].

Key-generation. Let (KG_0, Sig_0, Ver_0) and (KE_1, Sig_1, Ver_1) be two signature schemes that secure against adaptive chosen message attacks. On a security parameter n, each verifier V randomly generates two pair $(verk_0, sigk_0)$ and $(verk_1, sigk_1)$, where $verk$ is the signature verification key and $sigk$ is the signing key. V publishes $(verk_0, verk_1)$ as its public-key while keeping $sigk_b$ as its secret-key for a randomly chosen b from $\{0, 1\}$ (V discards $sigk_{1-b}$).

Common input. An element $x \in L$ of length $poly(n)$, where L is an $\mathcal{NP}$-language that admits Σ-protocols.

The main-body of the protocol. The main-body of the protocol consists of the following three phases:

Phase-1. The verifier V proves to P that it knows the either $sigk_0$ or $sigk_1$, by executing the Σ_{OR}-protocol on $(verk_0, verk_1)$ in which V plays the role of knowledge prover. It is additionally required that the first-round message of the Σ_{OR}-protocol is generated without using the knowledge of either $sigk_0$ or $sigk_1$ (i.e., *partial witness-independent*). Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol of this phase respectively. Here e_V is the random challenge sent by the prover to the verifier.

If V successfully finishes the Σ_{OR}-protocol of this phase and P accepts, then goto Phase-2. Otherwise, P aborts.

Phase-2. P generates a key pair (sk, vk) for a one-time strong signature scheme. Let COM be a commitment scheme. The prover randomly selects random strings $s, r \in \{0,1\}^n$, and computes $C = COM(s, r)$ (that is, P commits to s using randomness r). Finally, P sends (C, vk) to the verifier V. .

Phase-3. By running a Σ_{OR}-protocol, P proves to V that it knows either a witness w for $x \in L$ OR the value committed in C is a signature on the message of vk under either $verk_0$ or $verk_1$. Denote by a_P, e_P, z_P, the first-round, the second-round and the third-round message of the Σ_{OR} of Phase-3. Finally, P computes a signature δ on the whole transcript using one-time strong signature by using the signing key sk generated in Phase-2.

Verifier's decision. V accepts if and only if the Σ_{OR}-protocol of Phase-3 is accepting, and δ is a valid signature on the whole transcript under vk.

Note: The actual implementation of the DDL protocol combines rounds of the above protocol. But, it is easy to see that round-combination does not invalid the following attacks.

11.1.1 Concurrent MIM attack against right sessions

We first show a special CMIM attack in which the adversary A only participate the right concurrent interactions with honest verifiers (i.e., there are no concurrent left interactions in which A concurrently interacts with honest provers).

The following CMIM attack enables A to malleate the interactions of Phase-1 of one session into a successful conversation of another concurrent session for different (but verifier's public-key related) statements without knowing any corresponding $\mathcal{NP}$-witnesses.

Let $\hat{L}$ be any $\mathcal{NP}$-language admitting a Σ-protocol that is denoted by $\Sigma_{\hat{L}}$ (*in particular, $\hat{L}$ can be an empty set*). For an honest verifier V with its public-key $PK = (verk_0, verk_1)$, we define a new language $L = \{(\hat{x}, verk_0, verk_1) | \exists w \ s.t. \ (\hat{x}, w) \in R_{\hat{L}} \text{ OR } w = verk_b for \ b \in \{0,1\}\}$. Note that for any string $\hat{x}$ (whether $\hat{x} \in \hat{L}$ or not), the statement "$(\hat{x}, verk_0, verk_1) \in L$" is always true as $PK = (verk_0, verk_1)$ is honestly generated. Also note that L is a language that admits Σ-protocols (as Σ_{OR}-protocol is itself a Σ-protocol). Now, we describe the concurrent interleaving and malleating attack, in which A successfully convinces the honest verifier of the statement "$(\hat{x}, verk_0, verk_1) \in L$" for *any arbitrary poly*(n)-bit string $\hat{x}$ (*even when* $\hat{x} \notin \hat{L}$) by concurrently interacting with V (with public-key $(verk_0, verk_1)$)in two (right) sessions as follows (note that A does not need to participate concurrent left interactions).

1. A initiates the first session with V. After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the first session on common input $(verk_0, verk_1)$ (i.e., V's public-key), A suspends the first session.

2. A initiates a second session with V, and works just as the honest prover does in Phase-1 and Phase-2 of the second session. We denote by C, vk the Phase-2 message of the second session, where C the commit to a random string and vk is the verification key of the one-time strong signature scheme generated by A (*note that A knows the corresponding signing key sk as (vk, sk) is generated by itself*). When A moves into Phase-3 of the second session and needs to send V the first-round message, denoted by a_P, of the Σ_{OR}-protocol of Phase-3 of the second session *on common input* $(\hat{x}, verk_0, verk_1)$, A does the following:

   - A first runs the SHVZK simulator of $\Sigma_{\hat{L}}$ (i.e., the Σ-protocol for $\hat{L}$) on $\hat{x}$ to get a simulated conversation, denoted by $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$".

- A runs the SHVZK simulator of the Σ-protocol for showing that the committed value in C is a signature on vk under one of $(verk_0, verk_1)$ to get a simulated conversation, denoted by (a_C, e_C, z_C).
- A sets $a_P = (a_{\hat{x}}, a'_V, a_C)$ and sends a_P to V as the first-round message of the Σ_{OR}-protocol of Phase-3 of the second session, where a'_V is the one received by A in the first session.
- After receiving the second-round message of Phase-3 of the second session, i.e., the random challenge e_P from V. A suspends the second session.

3. A continues the first session, and sends $e'_V = e_P \oplus e_{\hat{x}} \oplus e_C$ as the second-round message of the Σ_{OR}-protocol of Phase-1 of the first session.

4. After receiving the third-round message of the Σ_{OR}-protocol of Phase-1 of the first session, denoted by z'_V, A suspends the first session again.

5. A continues the execution of the second session again, sends to $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V), (e_C, z_C))$ to V as the third-round message of the Σ_{OR}-protocol of the second session.

6. Finally, A applies sk on the whole transcript *of the second session* to get a (one-time strong) signature δ, and sends δ to V

Note that $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$ is an accepting conversation for the (possibly false) statement "$\hat{x} \in \hat{L}$", (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of either $sigk_0$ or $sigk_1$, (a_C, e_C, z_C) is an accepting conversation for showing that the value committed in C is a signature on vk under one of $(verk_0, verk_1)$. Furthermore, $e_{\hat{x}} \oplus e'_V \oplus e_C = e_P$, and δ is a valid (one-time strong) signature on the transcript of the second session.This means that, from the viewpoint of V, A successfully convinced V of the statement "$(\hat{x}, verk_0, verk_1) \in L$" in the second session *but without knowing any corresponding $\mathcal{NP}$-witness*! Also, A even does not need to participate concurrent left interactions. We remark that mixing the public key structure as part of the language is a natural attack strategy for the public-key model (a different demonstration of this was given in [71]).

Note: Although the above attack is described against the DDL ZK protocol of [27]. But, it can be trivially modified to be an attack against the CZK/RZK-CS protocol of [26], showing all the ZK protocols of [27, 26] are not CKE.

11.1.2 CMIM attack against left sessions

Using similar attack strategy as described above, an MIM adversary can also malleate left session interaction on a statement x into a successful conversation of right session on a related (different) statement x'. Specifically, using strategy similar to the above CMIM attack against left sessions, we can also give a CMIM attack that A concurrently interacts one left session and several (specifically, two) right sessions and can malleate the left session interaction on a common input $x \in L$ into a successful conversation of a right session on a new and maliciously related $\mathcal{NP}$-statement for the common input $(x, verk_0, verk_1, vk, C)$ in the right session, in which the MIM adversary proves to the honest verifier of the knowledge of either a witness for $x \in L$ or the value committed in C is signature of vk under either $verk_0$ or $verk_1$ *but without knowing any $\mathcal{NP}$-witness.* In more details, A replays all messages sent by honest verifier in one right session to the honest prover in the left session, denote by (vk, C) the Phase-2 message sent by P in the left session. Note that although the common input of P in the left session is x, BUT in Phase-3 of the left session P will give a Σ-protocol for showing either $x \in L$ or the value committed in C is a signature of vk under either $verk_0$ or $verk_1$. Then, A initiates a new right session and sets $(x, verk_0, verk_1, vk, C)$ as the common input of the new initiated right session. In phase-2 of

this right session, A generates (vk', sk', C'), sends (vk', C') to V. Also, A generates a simulated Σ-proof for showing C' commits to a signature of vk' under either $verk_0$ or $verk_1$. Then, using strategy similar to above attack against right sessions, A malleates the Phase-3 interaction of the left session into a successful Phase-3 interaction of the second right session as follows: it combines the real interaction of Phase-3 of the left session with the above simulated Σ-proof to form an accepting Σ_{OR} conversation of Phase-3 of the second right session, and applying sk' on the whole transcript of the second right session to get the one-time strong signature. Details are omitted here.

11.1.3 Comments

We note that the notion of concurrent non-malleability is formulated and motivated to prevent CMIM attacks. A protocol is formulated to be CNM secure, if it achieves the intuitive goal: concurrent left and right interactions does not help a CMIM adversary to convince an honest verifier of a statement for which it actually does not know a witness (this is in particular the goal and motivation clearly mentioned in [27]). But, the above attacks show that the DDL protocol does not achieve CNM security.

A careful investigation shows that there are subtle points both in the proof and in the security model and formulation in [27].

For security analysis of [27], in the simulation-extraction run by the simulator/extractor who accesses A has simulated secret-key. In the Claim-1 of Page-13, it is concluded that if Claim-1 does not hold, then the signature extracted will be w.r.t. the same signing key used by the simulator in all right sessions. In particular, it is claimed that the view of A in the left sessions is independent of which signing key used by the simulator in all right sessions. Also, in the security analysis, the simulator will uses one of the two signing keys randomly for any left session w.r.t to the same simulated public-key (the simulator knows both the two simulated signing keys).

We think the proof of Claim-1 in [27] is a bit problematic. Firstly, contrary to the claim that the view of A in all left sessions is independent of the signing key used by the simulator in all right sessions, we suggest it may not be this case. The reason is as follows: firstly, in the key-generation phase, A may have generated a list of public-keys that are maliciously related to the simulated public-key $(verk_0, verk_1)$. Secondly, in the main-proof stage, A can potentially malleate the proof of V on $(verk_0, verk_1)$ into a successful proof on its maliciously related public-key *but actually without knowing any secret-key*. This means that the secret-keys extracted by the simulator/extractor for left sessions w.r.t. public-keys generated by A may potentially be maliciously related to the signing-key used by the simulator in left sessions (e.g, they satisfy some malicious relations). Furthermore, the left sessions executed by the simulator with the extracted secret-keys can give A extra additional advantage, in the sense that the view of the left sessions cannot be "emulated" by A itself as it may actually not know the secret-keys corresponding to its maliciously generated public-keys. This means that the view of A in left sessions may still be potentially related to the signing-key used by the simulator in the right sessions. The security analysis of [27] seems not deal with this subtle case.

More importantly, subtleties exist as well in the security model and formulation. Details are presented in next subsection.

11.2 Formulation of concurrent non-malleability in the public-key model, revisited

Normal formulation of CNM in the standard model roughly is the following: for any PPT CMIM adversary A that takes part in both concurrent left sessions with honest provers and concurrent right sessions with honest verifier, there exist a PPT simulator/extractor E such that E outputs the following: (1). a simulated transcript that is indistinguishable from the real view of in its CMIM attacks. (2). For a successful right session on a common input x and the transcript of this right session is different from

those of all *RIGHT* sessions, E can output a corresponding witness.

The requirement (1) intuitively captures that any advantage of A can get by concurrent left and right interactions can also be got by E itself alone without any interactions, i.e, A gets no extra advantage by the CMIM attacks. The requirement (2) intuitively captures that for any statement that A convince of V in one of right sessions (with its transcript is different with those of all left sessions), A must "know" a witness.

Formulating CNM for *interactive* protocols in the public-key model only begins very recently (and actually started in the early version of this work). The formulation of CNM in the public-key model in all previous works (in particular, the works of [61, 27] and also the early version of this work) almost directly bring the CNM formulation (originally formulated for the standard setting) into the public-key setting, but with the following difference: E will simulate the key-generation phases of all honest verifiers. Put in another word, in its simulation/extration E actually takes the corresponding secret-keys of honest verifiers.

We highlight some key differences between the CMIM setting in the public-key model and the CMIM setting in the standard model.

1. For CMIM setting in the standard model, honest verifiers are PPT algorithms. In this case, normal CNM formulation only considers the advantages the CMIM adversary can get from concurrent left sessions (as the actions of honest verifiers in right sessions can be emulated perfectly); But, for CMIM setting in the public-key model, honest verifiers possesses secret value (i.e, its secret-key) that can *NOT* be computed efficiently. In other words, in this case an CMIM adversary can get advantages both from the left sessions and *from the right sessions*. This is a crucial difference between CMIM settings for standard model and public-key model that normal formulation of CNM does not capture. The above CMIM attack against right sessions clearly demonstrates this difference.

2. For CNM setting in the public-key model, the CMIM adversary can put public-keys maliciously related to public-keys generated by honest verifiers. Furthermore, for CNM setting in the bare public-key model, where dynamic interleaving key generation is allowed with some limitation (i.e., there is a reasonable amount time interval between the time of key registration and the time of key usage, more details see [13]), the public-keys generated by the CMIM adversary are not only maliciously related to public-keys generated by honest verifiers, more importantly, but also maliciously related to the whole transcript of completed and incompleted sessions in both concurrent left part and the concurrent right part. Previous works do not capture the late case, which is more common in practice.

With the above key differences in mind, we re-formulate the CNM notion in the public-key model. Most parts of the simulation can be inherited from the early version of this work (i.e., Section 6) and the works [61, 27] that all in turn are (incomplete) adaption of the the standard-model CNM formulation. Below, we only highlight the key differences of our formulation with previous formulations. Formal formulations can be easily got accordingly, and details are omitted here. In the following CNM formulation, we simply assume there is only one honest verifier (i.e., all right sessions are w.r.t the same honest verifier with the same public-key). Generalization to the case of multiple honest verifiers is straightforward.

Above all, in the knowledge-extraction, besides requiring the ability to output a simulated indistinguishable transcript, we require that for any accepting right session with transcript that is different with any other sessions (*not only left sessions*), as required in the formulation of CKE in Section 9, we explicitly mandate the knowledge extracted is "*independent*" of the (whether simulated or real)

secret-key used by E (for emulating honest verifier in the process of knowledge-extraction). There are three levels of independence to formulate:

- We require the $\Pr[R(w, SK)] = 1$ is negligibly close to $\Pr[R(w, SK')] = 1$ for any polynomial-time computable relation R, where SK' is some element *randomly and independently* distributed over the space of SK. This is a strong independence formulation, and thus may impose stronger complexity assumptions for protocols constructions. In particular, our CZK-CKE protocols presented in Section 9 can be implemented with any OWP or any OWF that has unique preimage for achieving this strong formulation of independence.

- We require the $\Pr[R(w, SK)] = 1$ is negligibly close to $\Pr[R(w, SK')] = 1$ for any polynomial-time computable relation R, where SK' is some element independently distributed over the space of SK. This is a relaxed independence formulation, and thus may allow weaker complexity assumptions for protocols constructions. In particular, our CZK-CKE protocols presented in Section 9 can be implemented with any OWF for achieving this strong formulation of independence.

- We require the $\Pr[R(w, SK)] = 1$ is negligibly close to $\Pr[R(w, SK')] = 1$ for any polynomial-time computable relation R, where SK' is some (different) element in the space of SK. This is a much relaxed independence formulation, and may potentially be problematic. The reason is that: SK' may be related to SK (under some relation), and in this case the above formulation may potentially be problematic. But in some applications, this relaxed formulation may render us efficient implementations, and is still useful in certain scenarios.

Remark: In the formal experiment used to define CNM in public-key model, we can let the simulator/extractor output SK' and explicitly require that the distribution of SK' is independent of that of SK. In particular, we can let the simulator/extractor first generates a triple $(PK, SK, SK'$ on the top of the experiment, where (PK, SK) is the simulated key-pair, and SK' is randomly and independently distributed over the space of SK. The common trick for protocol design that allows such CNM security is, of course, to let the verifier uses two independent public keys.

Secondly, we (optionally) allow the CMIM adversary can generate its keys based on the whole view dynamically, with the limitation imposed in [13] that there is a reasonable time interval between key registration and key usage. This requirement can be optional, as this requirement may be too strong to allow efficient implementations of CNM protocols in the public-key model.

Finally, we clarify and classify the ability of CMIM adversary A for adaptive input selection against $\mathcal{NP}$-protocols (i.e., protocols that work for any $\mathcal{NP}$-language.

For adaptive input selection for $\mathcal{NP}$-protocols, the formulations of CNM in the standard model or in setup models normally only consider the restricted case: The inputs (statements) selected by the CMIM adversary for both left sessions and right sessions are for *same* (predefined) $\mathcal{NP}$-relation. In full adaptive sections, A can select any NP-statements to be used in both left sessions and right sessions of the same NP-protocol. *We note that allowing full adaptive input selection is very reasonable to capture adversarial abilities of CMIM adversaries in practice, especially when NP-protocols are run concurrently over Internet that is just the scenario for which CNM is motivated. Especially, for $\mathcal{NP}$-protocols concurrently run in the public-key model, mixing public-key structures are a very natural and practical adversarial strategy in practice (thus we need to formulate them).* We remark that full adaptive input selection is different from concurrent general composition, which considers the scenario where *different* protocols are executed concurrently and thus with different NP-statements. For CNM protocol with full adaptive input selection, we consider that the *same* NP-protocol are executed concurrently but the common inputs to those concurrent sessions may not be for the same (predefined) $\mathcal{NP}$-relation. Thus, our formulation of CNM with full adaptive input selection can be a hybrid between traditional

CNM and the concurrent general composition. Again, there are three level adaptive input selection to formulate for NP-protocols:

- Full adaptive input selection: in which A can select any NP-statements for any left or right sessions.
- Semi adaptive input selection: in which A can select NP-statements for both left and right sessions but with the restriction that the statements selected for left sessions are subject to a predefined $\mathcal{NP}$-relation.
- Predefined adaptive input selection: in which all statements selected by A are subject to the same predefined NP-relation.

Comments: Note that the first CMIM attack against the DDL protocol shows that it is not CNM w.r.t. predefined adaptive input selection; and the second attack shows that it is not CNM w.r.t semi adaptive input selection. Similar but different formulation of adaptive input selection is also presented in [61]. Here, we clarify the differences between our formulation and the formulation of [61]. The work of [61] classifies the input selection ability according to which part the adversary can select inputs, and the formulation of [61] implicitly assumes that the NP-statements selected are for the same NP-relation. Informally, the full adaptive input selection formulated in [61] corresponds to the above predefined adaptive input selection of our formulation.

11.2.1 Observations on the CNM formulation of [61]

Our above CNM formulation is the simulation-based formulation. Recently, the works [63] investigates the indistinguishability-based approach for formulating CNM for interactive protocols, e.g., ZK and commitment schemes. The CNM formulation in [61] follows the indistinguishability-based formulation approach of [62, 63] originally formulated for the standard model. Specifically, in the CNM formulation of [61] two experiments are defined (page 19): a real experiment w.r.t. a real public-key of an honest verifier (here, denoted PK_V), in which a CMIM adversary mounts CMIM attacks; a simulated experiment run by a simulator/extractor S w.r.t. a simulated public-key (here, denoted PK_S), in which S accesses A and takes a simulated secret-key SK_S. The CNM is then formulated as follows: the distributions of all witness used by A in right sessions in the real experiment is indistinguishable from the distribution of the witness used by A in right sessions in the simulated experiment. Note that [61] does not require the simulator/extractor to output a simulated transcript.

It appears that the CNM formulation of [61] has already dealt with the issue of knowledge-extraction independence. Note that our attack-driven CKE formulation presented in Section 9 (on 19 Aug., 2006) and the CNM formulation of [61] are totally independent. But, the key observation here is that the CNM formulation in [61] is also incomplete, and insecure protocols could potentially be "proved" secure w.r.t. the CNM formulation of [61]. The reason is as follows:

Firstly, in the real experiment the statements selected by the CMIM adversary A for both left and right sessions can be maliciously related to PK_V (e.g., some function of PK_V), and thus the witness extracted for right sessions of the real experiment could be potentially dependent on the secret-key SK_V used by honest provers and the honest verifier (note that SK_V could be given to honest provers as witness in left sessions). Note that, as witnessed by our concrete CMIM attacks, when extracted witness are maliciously dependent on SK_V knowledge-extraction does not fully capture the intuition that V does know the witness extracted (specifically, A may actually not know the extracted witnesses). Similarly, as in the simulated experiment S uses SK_S in simulation/extraction, the witness used by A in right sessions in the simulated experiment could also be maliciously dependent on SK_S. That is,

both the witness used by A in real experiment and in the simulated experiment may be maliciously dependent on SK_V and SK_S respectively, *but the distributions of them still can be indistinguishable as the distributions of SK_V and SK_S are identical!*. The natural augmentation is to explicitly require knowledge-extraction independence for the simulated experiment in accordance with the three level of independence interpretations.

Comments: A salient feature of our formulation of CNM and CKE in public-key model, in comparison with other CNM formulations in the public-key model, is that our formulation is driven by concrete attacks. Also, we note that although the CNM formulation of [61] is incomplete, but it does not mean the OPV ZK protocol in the public-key model is not secure. Indeed, we do not know attacks on the OPV protocol, and actually the OPV protocol seems to be CNM secure. The key point here is: a proof according to the CNM formulations of [61, 27] does not necessarily guarantee CNM security in the public-key model, and correct formulations of CNM in the public-key model are far more complex and subtle than one first glance. In any case, the concrete attacks and the clarifications presented in this work play a critical role.

11.3 On achieving black-box CNMZK in the BPK model

Firstly, we note that no previous ZK protocols can be proved secure in the BPK model according to our above CNM formulation. The CNMZK protocol of [61] seems to be CNM secure with non-interleaving key registration, although [61] did not notice the subtleties in formulating CNM in the public-key model (that are driven by our concrete attacks).

We then investigate our CZK-CKE protocol presented in Section 9. It is easy to show that the CMIM attacks against DDL ZK protocol of [27] do *NOT* work on our CZK-CKE protocol presented in Section 9. But, we do not know whether it can be provable CNM secure.

11.4 Suggestions for fixing DDL protocol to against CMIM attacks

The first fix, inspired by our CZK-CKE protocol, is to require the prover of the DDL protocol to first commit to the NP-witness for $x \in L$. Specifically, in the fixed DDL protocol, as in our CZK-CKE protocol the prover sends two commitments in Phase-2, and then proves to the verifier that either the value committed in the first commitment is a witness for $x \in L$ OR the value committed in the second commitment is a signature on vk under either $verk_0$ or $verk_1$. We call this fixed protocol as DDL-CKE.

Another fixing that seems to achieve stronger security against CMIM attack is to further augment DDL-CKE by requiring the prover first commit to, in Phase-2, the first-round message of the Σ-protocol of Phase-3 of DDL-CKE using a simulation-sound trapdoor commitment (SSTC) (or other forms of commitments like multi-trapdoor commitments, CNM commitments in the common reference string model). In the last round, the prover decommits. In this case, the verifier's public-key consists of commitment public-keys (or common reference string if we use CNM commitment in the common reference string model). For provable security, we require the verifier proves the knowledge of commitment trapdoor key (again, we may need to use the key-pair trick). To prevent CMIM attacks against the proofs given by verifiers in Phase-1, similar fixing can also be applied to the Σ-protocol of Phase-1. In the later case, the prover also registeres public-key. We call such Σ-protocols as committed Σ-protocol, and such fixed protocols as Committed-DDL-protocol.

We stress that the above fixing approaches are ad hoc. Although they seem to provide seemingly stronger security against potential CMIM attacks (and clearly frustrate the know developed attacks), but we do not know how to formally prove the security according to our formulation of CNM in the public-key model.

11.5 Extension

We remark that although we revisited the issue of CNM formulation in the bare public-key model and for ZK, the arguments can also be easily extended to be applied to other public-key models and to other primitives like commitments, oblivious transfer, general secure multiparty computation et al., and to the formulation of more advanced notion like concurrent general composition CGC and universal composition UC. In all these cases, the principle is that we should explicitly mandate that the knowledge-extracted should be independent of the (whether simulated or real) secret-keys.

References

[1] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *IEEE Symposium on Foundations of Computer Science*, pages 106-115, 2001.

[2] B. Barak. Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In *IEEE Symposium on Foundations of Computer Science*, pages , 2002.

[3] B. Barak and O. Goldreich. Universal Arguments and Their Applications. In*IEEE Conference on Computational Complexity*, pages 194-203, 2002.

[4] B. Barak, O. Goldreich, S. Goldwasser and Y. Lindell. Resettably-Sound Zero-Knowledge and Its Applications. In *IEEE Symposium on Foundations of Computer Science*, pages 116-125, 2001

[5] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. In *ACM Symposium on Theory of Computing*, pages 484-493, 2002.

[6] M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key-Exchange Protocols. In *ACM Symposium on Theory of Computing*, pages 419-428, 1998.

[7] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1992, LNCS 740*, pages 390-420. Springer-Verlag, 1992.

[8] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Asiacrypt'00. LNCS 1976, Springer-Verlag, 2000. (authentication does not implies non-malleability).

[9] M. Blum. Coin Flipping by Telephone. In *proc. IEEE Spring COMPCOM*, pages 133-137, 1982.

[10] M. Blum. How to Prove a Theorem so No One Else can Claim It. In Proceedings of the International Congress of Mathematicians, Berkeley, California, USA, 1986, pp. 1444-1451.

[11] Brassard, D. Chaum and C. Crepeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer Systems and Science*, 37(2): 156-189, 1988.

[12] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136-145, 2001.

[13] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 235-244, 2000.

[14] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(log\ n)$ Rounds. In *ACM Symposium on Theory of Computing*, pages 570-579, 2001.

[15] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds. In *SIAM Journal on Computing*, 32(1): 1-47, 2002.

[16] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *ACM Symposium on Theory of Computing*, pages 494-503, 2002.

[17] R. Cramer and I. Damgard. On Electronic Payment Systems. A lecture note for the course of Cryptographic Protocol Theory at Aarhus University, 2003. Available from: http://www.daimi.au.dk/∼ivan/CPT.html

[18] R. Cramer, I. Damgard and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Y. Desmedt (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1994, LNCS 839*, pages 174-187. Springer-Verlag, 1994.

[19] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *B. Preneel (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2000, LNCS 1807*, pages 418-430. Springer-Verlag, 2000.

[20] I. Damgard. On Σ-protocols. A lecture note for the course of Cryptographic Protocol Theory at Aarhus University, 2003. Available from: http://www.daimi.au.dk/∼ivan/CPT.html

[21] I. Damgard and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *ACM Symposium on Theory of Computing*, pages 426-437, 2003.

[22] I. Damgard and J. B. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In *M. Yung (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 581-596. Springer-Verlag, 2002.

[23] I. B. Damgard, T. P. Pedersen and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. *Journal of Cryptology*, 10(3): 163-194, 1997.

[24] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-Interactive Zero-Knowledge. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 566-598. Springer-Verlag, 2001.

[25] A. De Santis, G. Di Crescenzo and G. Persiano. Zero-Knowledge Arguments and Public-Key Cryptography. Information and Computation. 121(1): 23-40 (1995)

[26] Y. Deng and D. Lin. Resettable Zero Knowledge in the Bare Public-Key Model under Standard Assumption. Cryptology ePrint Archive, Report No. 2006/239, July 12, 2006.

[27] Y. Deng, G. Di Crescenzo, and D. Lin. Concurrently Non-Malleable Zero-Knowledge in the Authenticated Public-Key Model. Cryptology ePrint Archive, Report No. 2006/239, September 12, 2006.

[28] G. Di Crescenzo, G. Persiano and I. Visconti. Constant-Round Resettable Zero-Knowledge with Concurrent Soundness in the Bare Public-Key Model. In *M. Franklin (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2004, LNCS 3152*, pages 237-253. Springer-Verlag, 2004.

[29] G. Di Crescenzo and I. Visconti. Concurrent Zero-Knowledge in the Public-Key Model. In *L. Caires et al. (Ed.): ICALP 2005, LNCS 3580*, pages 816-827. Springer-Verlag, 2005.

[30] G. Di Crescenzo, Y. Ishai and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment In *ACM Symposium on Theory of Computing*, pages 141-150, 1998.

[31] G. Di Crescenzo, J. Katz, R. Ostrovsky and A. Smith. Efficient and Non-Interactive Non-Malleable Commitments. In *B. Pfitzmann (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, pages 40-59. Springer-Verlag, 2001.

[32] G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Pre-Processing. In *M. J. Wiener (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1999, LNCS 1666*, pages 485-502. Springer-Verlag, 1999.

[33] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. In *ACM Symposium on Theory of Computing*, pages 542-552, 1991.

[34] C. Dwork and M. Naor. Zaps and Their Applications. In *IEEE Symposium on Foundations of Computer Science*, pages 283-293, 2000. Available on-line from:

[35] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 409-418, 1998.

[36] T. El Gamal. A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31: 469-472, 1985.

[37] U. Feige. Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990. Available from: `http://www.wisdom.weizmann.ac.il/~feige`.

[38] U. Feige, A. Fiat and A. Shamir. Zero-knowledge Proof of Identity. *Journal of Cryptology*, 1(2): 77-94, 1988.

[39] U.Feige, D. Lapidot and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, 29(1): 1-28, 1999.

[40] U. Feige and Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *G. Brassard (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1989, LNCS 435*, pages 526-544. Springer-Verlag, 1989.

[41] M. Fischlin and R. Fischlin. Efficient Non-Malleable Commitment Schemes. In *M. Bellare (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2000, LNCS 1880*, pages 413-431. Springer-Verlag, 2000.

[42] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *A. Odlyzko (Ed.): Advances in Cryptology-Proceedings of CRYPTO'86, LNCS 263*, pages 186-194. Springer-Verlag, 1986.

[43] J. A. Garay, P. MacKenzie and K. Yang. Strengthening Zero-Knowledge Protocols Using Signatures. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 177-194. Springer-Verlag, 2003.

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

[45] O. Goldreich. Concurrent Zero-Knowledge with Timing, Revisited. In *ACM Symposium on Theory of Computing*, pages 332-340, 2002.

[46] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game-A Completeness Theorem for Protocols with Honest Majority. In *ACM Symposium on Theory of Computing*, pages 218-229, 1987.

[47] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All language in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery*, 38(1): 691-729, 1991.

[48] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof System. *SIAM Journal on Computing*, 18(1): 186-208, 1989.

[49] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes From Collision-Free Hashing. In *N. Koblitz (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, pages 201-215. Springer-Verlag, 1996.

[50] J. Kilian and E. Petrank. Concurrent and Resettable Zero-Knowledge in Poly-Logarithmic Rounds. In *ACM Symposium on Theory of Computing*, pages 560-569, 2001.

[51] J. Kilian, E. Petrank, R. Richardson. Concurrent Zero-Knowledge Proofs for $\mathcal{NP}$. Available from: http://www.cs.technion.ac.il/~erez/Papers/czkub-full.ps.

[52] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology-Proceedings of CRYPTO 2005, LNCS 3621*, pages 546-566. Springer-Verlag, 2005. Full version appears in Cryptology ePrint Archive Report No. 2005/176.

[53] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 171-189. Springer-Verlag, 2001.

[54] Y. Lindell. A Simple Construction of CCA2-Secure Public-Key Enryption Under General Assumptions. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 241-255. Springer-Verlag, 2003.

[55] Y. Lindell. Composition of Secure Multi-Party Protocols - A Comprehensive Study. LNCS 2815, Springer-Verlag, 2003.

[56] A. Menezes. Another Look at HMQV. Cryptology ePrint Archive, Report No. 2005/205.

[57] D. Micciancio and E. Petrank. Simulatable Commitments and Efficient Concurrent Zero-Knowledge. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 140-159. Springer-Verlag, 2003.

[58] S. Micali and L. Reyzin. Soundness in the Public-Key Model. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 542–565. Springer-Verlag, 2001.

[59] S. Micali and L. Reyzin. Min-Round Resettable Zero-Knowledge in the Public-Key Model. In *B. Pfitzmann (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, pages 373–393. Springer-Verlag, 2001.

[60] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *ACM Symposium on Theory of Computing*, pages 427-437, 1990.

[61] R. Ostrovsky, G. Persiano and I. Visconti. Concurrent Non-Malleable Witness Indistinguishability and Its Applications. Cryptology ePrint Archive, Report No. 2006/256 (appears in early of August, 2006).

[62] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *ACM Symposium on Theory of Computing*, pages 533-542, 2005.

[63] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. In *IEEE Symposium on Foundations of Computer Science*, pages 563-572, 2005.

[64] M. Prabhakaran, A, Rosen and A. Sahai. Concurrent Zero-Knowledge With Logarithmic Round Complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 366-375, 2002.

[65] R. Richardson and J. Killian. On the Concurrent Composition of Zero-Knowledge Proofs. In *J. Stern (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1999, LNCS 1592*, pages 415-423. Springer-Verlag, 1999.

[66] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen Ciphertext Security. In *IEEE Symposium on Foundations of Computer Science*, pages 543-553, 1999.

[67] C. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3): 24, 1991.

[68] A. C. Yao. How to Generate and Exchange Secrets. In *IEEE Symposium on Foundations of Computer Science*, pages 162-167, 1986.

[69] Y. Zhao. Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model and Its Applications Unpublished manuscript, appears in Cryptology ePrint Archive Report No. 2003-265.

[70] M. Yung and Y. Zhao. Constant-Round Concurrently-Secure rZK with (Real) Bare Public-Keys. Electronic Colloquium on Computational Complexity, 12(48), 2005.

[71] M. Yung and Y. Zhao. Interactive Zero-Knowledge with Restricted Random Oracles. TCC 2006, to appear.

[72] Y. Zhao. Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model and Its Applications. Unpublished manuscript, appears in Cryptology ePrint Archive, Report 2003/265.

[73] Y. Zhao, X. Deng, C. H. Lee and H. Zhu. Resettable Zero-Knowledge in the Weak Public-Key Model. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 123-140. Springer-Verlag, 2003.

[74] Y. Zhao, J. B. Nielsen, R. Deng and D. Feng. Generic yet Practical ZK Arguments from any Public-Coin HVZK. Electronic Colloquium on Computational Complexity, 12(162), 2005.

[75] Y. Zhao, X. Deng, C. H. Lee and H. Zhu. Resettable Zero-Knowledge in the Weak Public-Key Model. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 123-140. Springer-Verlag, 2003.