Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model *

Yunlei Zhao†

Abstract

In this paper, we present both practical and general 4-round concurrent and resettable zero-knowledge arguments with *concurrent soundness* in the bare public-key (BPK) model. To our knowledge, our result is the first work that achieves concurrent soundness for ZK protocols in the BPK model and stands for the current state-of-the-art of concurrent zero-knowledge with setup assumptions. Since the BPK model is very simple and also very reasonable and 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, we suggest that zero-knowledge protocols with simultaneous concurrent security in the BPK model may be of independent interests and can be used as a building block in other applications in the BPK model (e. g. secure two-party and multi-party computation with registered public-keys).

Keywords: concurrent soundness, concurrent zero-knowledge, bare public-key (BPK) model

1 Introduction

The notion of zero-knowledge (ZK) was introduced in the seminal paper of Goldwasser, Micali and Rackoff [35] 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 [34] 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.

With the emergence and far and wide sweeping popularity of the Internet, much recent research attention, initiated by Dwork, Naor and Sahai [26], has been paid on the security threats of zero-knowledge protocols when 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. 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 controlling all the verifiers can schedule all the executions concurrently at its wish.

Although non-constant-round black-box concurrent zero-knowledge proofs for $\mathcal{NP}$ exist in the plain model under standard intractability assumptions [45, 38, 44], but they cannot be constant-round in the black-box sense [11]. Constant-round non-black-box bounded concurrent zero-knowledge arguments and arguments of knowledge for $\mathcal{NP}$ are achieved in [1, 4]. To achieve constant-round black-box concurrent

*Partial contents of this works appear in an on-going application for US patents.

†Department of Computer Science, Fudan University, Shanghai, P. R. China.
csylzhao@cityu.edu.hk

zero-knowledge, several computational models are introduced: the timing model [26], the preprocessing model [17] and the bare public-key model [10].

The bare public-key (BPK) model is first introduced by Canetti, Goldreich, Goldwasser and Micali [10] to achieve round-efficient resettable zero-knowledge (rZK) that is a generalization and strengthening of the notion of concurrent zero-knowledge. But as stated by Micali and Reyzin [41], 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) 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 BPK model is 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 rZK protocols exist both in the standard and in the BPK models [10], only in the latter case can they 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 [41]. 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. 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 even P^* is allowed multiple interleaved interactions with V. Resettable zero-knowledge protocols with concurrent soundness are really desirable in most smart-card based applications over Internet. Unfortunately, up to now we do not know how to construct resettable zero-knowledge protocols with concurrent soundness for $\mathcal{NP}$ in the BPK model and it is suggested as an interesting and important open problem in [10, 41]. Resettable zero-knowledge with concurrent soundness in some stronger version of the BPK model can be found in [42, 49].

1.1 Our contributions

In this paper, we present a constant-round (specifically, 6-round) black-box concurrent zero-knowledge argument of knowledge with *concurrent soundness* for $\mathcal{NP}$ in the bare public-key (BPK) model under standard intractability assumptions. To our knowledge, our result is the first work that achieves simultaneous concurrent security for zero-knowledge protocols in the BPK model. Note that the rZK protocols of [10, 41] guarantee concurrent (and more strongly, resettable) zero-knowledge (for concurrent prover security) but do not guarantee concurrent soundness (for concurrent verifier security). Furthermore, the works of [10, 41] are under sub-exponential hardness assumptions. In comparison, our work provides both concurrent prover security and concurrent verifier security for zero-knowledge protocols in the BPK model under standard intractability assumptions without assuming any sub-exponential or super-polynomial hardness assumption. Our protocol is not resettable zero-knowledge because any argument of knowledge protocol cannot be resettable zero-knowledge. Since the BPK model is very simple and also very reasonable and 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, we suggest that zero-knowledge protocols with simultaneous concurrent security in the BPK model may be of independent interests and can be used as a building block in other applications in the BPK model (e. g. secure two-party and multi-party computation with registered public-keys).

1.2 Organization

In Section 2, we quickly recall the major cryptographic tools used. In Section 3, we present the definitions of concurrent zero-knowledge and concurrent soundness in the BPK model. In Section 4, we present the constant-round black-box concurrent zero-knowledge argument of knowledge with concurrent soundness for $\mathcal{NP}$ in the BPK model and prove its security under standard intractability assumptions.

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 [14]. 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 [29] and efficient electronic payment systems [13]. For a good survey of Σ-protocols and their applications, readers are referred to [18, 13].

Definition 2.1 (Σ-protocol [14]) *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 [8]: 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 [**?**], which is based on the zero-knowledge proof for DHC (directed Hamiltonicity cicle) of Blum [**?**].

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 [47] 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 [14], 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 [18] *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 [31]. More recent advances of zero-knowledge arguments of knowledge can be found in [39, 4].

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 [7]. We remark that 2-round public-coin WI proofs for $\mathcal{NP}$ do exist under the existence of one-way permutations [25].

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 [27]. An efficient implementation based on any one-way permutation is presented in [37] and readers are referred to [22] for recent advances of NIZK.

Definition 2.6 (NIZK proof of knowledge [23]) *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 [23].

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

In this section, we present the main result of this paper: a constant-round (specifically, 6-round) concurrent zero-knowledge argument of knowledge with concurrent soundness for $\mathcal{NP}$ in the BPK model, $< P, V >$, which is depicted in Figure 1 (page 8).

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 [22]) 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 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 [10] 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 21).

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 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

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

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. □

5 Improvements

In this section, we present two implementations for 4-round (that is optimal) black-box concurrent zero-knowledge argument of knowledge that enjoys both concurrent soundness and concurrent black-box witness extraction in the BPK model: A practical construction that is under the DLP assumption and runs in six rounds without going through general $\mathcal{NP}$-reductions for any language that admits Σ-protocols; And a general construction that runs in optimal rounds (4-round) for any language in $\mathcal{NP}$ under only one-way functions. To our knowledge, both the practical protocol and the general protocol stand for the current state-of-the-art of concurrent zero-knowledge with setup assumptions.

5.1 The Practical construction

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 14). 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 5.1 *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 [10] 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 17).

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. □

5.2 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 subsection.

5.3 The General Construction

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 [1]. As usual, the general construction goes through $\mathcal{NP}$-reductions. The general protocol is depicted in Figure 5 (page 19)[2].

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 [41] 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

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

[2] The general protocol is actually the Feige-Shamir constant-round zero-knowledge protocol for $\mathcal{NP}$ [**?**] 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 [**?**] 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.

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 5.2 *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 20). 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. □

6 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[3].

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.

6.1 Proof Outlines

The standard proof techniques for rZK presented in [10] 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 [10].

[3]We remark that before I made this observation, Di Crescenzo, Persiano and Visconti informed me that they had achieved rZK-CS protocol in the BPK model under superpolynomial hardness assumptions. The works of Di Crescenzo et al. on rZK-CS are 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.

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 [10, 41], 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.

7 Applications

7.1 Cryptographic Protocols with Full Concurrent Security in the BPK Model

A major application of CZK-CS protocols presented in this paper is to be used as building blocks to achieve cryptographic protocols with not only simultaneous concurrent player security but also concurrent channel security (concurrent non-malleability) in the BPK model. Specifically, the practical protocol and the OWF-based general protoco can be used as building blocks to achieve respectively practical (without going NP reduction) and general OWF-based coin-tossing protocol with both concurrent player security and concurrent channel security in the BPK model, which can be in turn used to achieve ZK and commitments with both concurrent player security and concurrent channel security in the BPK model. These result will appear in another separate report very soon.

7.2 Practical rZK with Secret-Keys

Works presented in this subsection are partial contents in an ongoing US patent application.

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.

7.2.1 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 [10]. 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.

7.2.2 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, 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.

Acknowledgments. The author is deeply indebted to Yehuda Lindell for his deep comments on motivations, numerous kindly clarifications and very valuable discussions and suggestions. Some critical idea presented in this manuscript comes from Lindell although he declined to serve as a coauthor of this paper. In particular, the 4-round CZK with councurrent soundness under only OWF is suggested by Lindell in January 2004. The author is full of gratitude to Di Crescenzo, Persiano and Ivan Visconti for many very valuable discussions and helps. The author is grateful to Boaz Barak, Oded Goldreich, Wenbao Mao and Leonid Reyzin for their kindly clarifications and warm helps.

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

[5] 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.

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

[7] M. Blum. How to Prove a Theorem so No One Else can Claim It. In Proceedings of the International Congress of Mathematicians, 1 (2): 1444-1451, 1987.

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

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

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

[11] 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.

[12] 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.

[13] 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

[14] 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.

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

[16] G. D. 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.

[17] G. D. 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.

[18] 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

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

[20] 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.

[21] 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.

[22] A. D. Santis, G. D. 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.

[23] A. D. Santis and G. Persiano. Zero-Knowledge Proofs of Knowledge Without Interaction. In *IEEE Symposium on Foundations of Computer Science*, pages 427-436, 1992.

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

[25] 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: http://www.wisdom.weizmann.ac.il/∼naor/

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] O. Goldreich and H. Krawczky. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1): 169-192, 1996.

[33] 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.

[34] 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.

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

[36] 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.

[37] J. Kilian, E. Petrank. An Efficient Non-Interactive Zero-Knowledge Proof System for $\mathcal{NP}$ with General Assumptions. *Journal of Cryptology*, 11(2): 24, 1998.

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

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

[49] 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.