SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2019, Vol. 62 032110:1–032110:14
https://doi.org/10.1007/s11432-018-9627-x

A novel approach to public-coin concurrent zero-knowledge and applications on resettable security

Zhenbin YAN[1,2*] & Yi DENG[1,2*]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China*

Received 1 May 2018/Accepted 7 October 2018/Published online 29 January 2019

Abstract Canetti, Lin and Paneth in TCC 2013 showed a $O(\log^{1+\varepsilon} n)$ rounds public-coin concurrent zero-knowledge argument system (CZK) based on the existence of collision resistant hash functions, which is currently known as round optimal public-coin CZK from standard assumptions. In this paper, we further address this problem and present an alternative construction of public-coin CZK argument system with succinct slot. The key technique involves a new variant of Barak's non-black-box simulate approach. In particular, the original protocol uses n commitments in each slot, while our construction uses one commitment in each slot. Through our simulation techniques, the simulator recovers any previous state needed for the probabilistically checkable proof (PCP) from the current committed state, which, in our view, may be of independent interest. Furthermore, the public-coin CZK argument system can be transformed into a resettable security protocol based on the one way functions assumption. Therefore, we present a new construction of the simultaneous resettable zero-knowledge argument system.

Keywords zero-knowledge, concurrent zero-knowledge, resettable zero-knowledge, concurrent secure computation, computational complexity

Citation Yan Z B, Deng Y. A novel approach to public-coin concurrent zero-knowledge and applications on resettable security. Sci China Inf Sci, 2019, 62(3): 032110, https://doi.org/10.1007/s11432-018-9627-x

1 Introduction

An interactive proof protocol is zero-knowledge if the prover can convince the verifier that a statement is true but does not reveal any information other than the fact itself. For example, a zero-knowledge protocol for the language $L \in$ NP will not reveal the witness w of the statement $x \in L$. Goldwasser et al. [1] formalized the definition by demonstrating that for every adversarial verifier V^*, there exists a simulator M that can simulate the entire interaction between the honest prover and verifier without having the witness. In the field of cryptography, zero-knowledge has been widely used in many security protocols.

When the zero-knowledge protocol is considered to run in an asynchronous network, Dwork et al. [2] introduced the notation of concurrent zero-knowledge (CZK) and demonstrated a first construction based on the timing assumptions. Concurrent zero-knowledge protocol is the key building block that can be used to construct more complex concurrently secure protocol for functionalities such as resettable security [3–6], secure multiparty computation [7–9], concurrent secure computation [10, 11], universally composable protocol [12–14], and so on.

* Corresponding author (email: yanzhenbin@iie.ac.cn, deng@iie.ac.cn)

© Science China Press and Springer-Verlag GmbH Germany, part of Springer Nature 2019

Based on the standard assumptions, Richardson et al. [15] constructed a black-box CZK argument protocol with $O(n^{\epsilon})$ rounds complexity. Canetti et al. [16] gave a proof that the round complexity of black-box CZK argument protocol is at least $\widetilde{\Omega}(\log n)$ rounds. Present, round-optimal construction of black-box CZK was proposed by Prabhakaran et al. [17] with $\widetilde{O}(\log n)$ rounds complexity. The construction is based on the structure of the Goldreich-Kan ZK protocol [18]. Pass et al. [19] recently developed a new construction with the same round complexity based on the structure of the Feige-Shamir ZK protocol [20].

However, the zero knowledge protocols above are not public-coin. The public-coin means that the honest verifier just needs to send messages in the form of uniform random coins that are independent of the entire history. A breakthrough was achieved through the work of Barak [21]. They demonstrated a "straight-line" non-black-box simulator by using the description of the verifier as trapdoor and constructed a first constant-round public-coin ZK protocol and bounded public-coin CZK protocol. Using a variant of [21], Pass et al. [22] demonstrated a constant-round parallel public-coin ZK argument protocol.

In the public-coin fully concurrent setting, Canetti et al. [23] used a "special-purpose" universal argument to achieve a $O(\log^{1+\varepsilon} n)$ rounds public-coin non-black-box CZK argument system by assuming the existence of collision resistant hash functions (CRHFs). In their paper [23], they need the hash function $h \in \mathcal{H}$ can be accessed by all the parties in a global model. Along this research route, Chung et al. [24] showed a public-coin CZK argument system based on the P-certificates assumption. Their construction is a constant-round implementation and is based on the variant of Barak's protocol. However, their protocol can only achieve uniform computational soundness. Following this, Goyal [25] achieved a public-coin CZK argument system by assuming the existence of CRHFs in the plain model. To ensure that the simulator extracts the trapdoor, they used several techniques in combinatorial mathematics and developed a protocol with n^{ϵ} round complexity. Based on [25], Kiyoshima [26] presented a new approach on the construction of n^{ϵ} rounds public-coin CZK argument system, and their result is also based on the CRHFs assumption.

Very recently, Pandey et al. [27] constructed a constant-round CZK argument protocol based on the $di\mathcal{O}$ assumption, and then Chung et al. [28] constructed a constant-round CZK argument protocol based on the $i\mathcal{O}$ assumption. Unfortunately, the last two protocols are not public-coin.

Our results. In this paper, we present a new variant of Barak's non-black-box simulation technique and give an improved construction on public-coin CZK argument system. In particular, our protocol only requires $O(\log^{1+\varepsilon} n)$-round complexity and each slot is "succinct" compared with [23]. We refer to a slot a "succinct slot" if each slot only contains one commitment c and challenge random string r (i.e., (c, r)) as the traditional Barak's protocol.

Theorem 1. There exists an $O(\log^{1+\varepsilon} n)$-round public-coin concurrent zero-knowledge argument system with "succinct slot" based on the existence of collision-resistant hash function families.

Recall that previous constructions in [23] were implemented in three stages. The first stage includes multiple-round commitment-challenge slots; the second stage includes a "special-purpose" universal argument to argue that the prover has a trapdoor for some slot in stage 1, and the last stage is a witness indistinguishable argument of knowledge (WIPOK) for proving either the prover knows a witness for statement $x \in L$ or the opening of the transcript in stage 2 is an accepted proof for knowing trapdoor of some slot. For optimization to $O(\log^{1+\varepsilon} n)$ rounds, they used two languages in stage 2, and require sending n commitments at each round in stage 1. However, we observe that this form is very complex and different from the traditional Barak's protocol, which only requires one commitment at each preamble slot. Through careful analysis, we provide a new construction with succinct slot and only require one commitment as the traditional Barak's protocol. Table 1 compares the characteristics of our protocol with other public-coin concurrent zero-knowledge protocols that were previously discussed.

2 Technical overview

In this section, we first recall the protocols achieved in [21, 23], then give a high-level description of our improved techniques.

Table 1 Comparison with recent results

Paper	Public-coin	Assumption	Round complexity	Succinct slot
Canetti et al. [23]	Yes	Global hash	$O(\log^{1+\epsilon}(n))$	No
Chung et al. [24]	Yes	CRHFs and P-certificate	$O(1)$	Yes
Goyal [25]	Yes	CRHFs	$O(n^{\epsilon})$	Yes
Kiyoshima [26]	Yes	CRHFs	$O(n^{\epsilon})$	No
This paper	Yes	Global hash	$O(\log^{1+\epsilon}(n))$	Yes

2.1 The protocol of [21]

Barak's protocol (P, V) is a public-coin zero-knowledge interactive protocol for NP. More specifically, let $L \in \text{NP}$ and R_L be a witness relation for L, then for $x \in \{0,1\}^n \cap L$, $w \in R_L(x)$, the description is given as follows:

Stage 1. V chooses $h \xleftarrow{R} \mathcal{H}_n$ and sends h to P.

Stage 2. P and V generate a slot (c, r):

(1) The honest prover P commits to dummy string 0^n, denotes $c \leftarrow \text{Com}(0^n)$ and sends c to V;

(2) The verifier V responds a random coins $r \in \{0,1\}^{2n}$ to P.

Stage 3. P runs a WIUA (witness-indistinguishable universal argument system) to prove the following OR statement:

(1) $\exists w$ s.t. $(x, w) \in R_L$ or

(2) $\exists M$ such that $M(c) = r$ and $c = \text{Com}(h(M))$, where the total time spent by M is bounded by $n^{\log\log n}$ steps.

To prove the soundness of the above protocol, roughly speaking, assume that there exists a cheating prover P^* who commits some program M', and sends $c = \text{Com}(h(M'))$. Because the prover does not have the code description M of the verifier, the probability $\Pr[M = M'] = \text{negl}(n)$. Now, for a fixed c of length n, let $M(c) = r, M'(c) = r'$, then $\Pr[M(c) = M'(c)] = \Pr[r = r'] = 2^{-2n}$. Because there are at most 2^n different c, the total probability of $\Pr[M(c) = M'(c)] = 2^{-n}$. Therefore, if P^* can cheat the verifier, then there must exists $M \neq M'$ but $c = \text{Com}(h(M')) = \text{Com}(h(M'))$ except with negligible probability. Therefore, we can use P^* to break the collision resistant security of CRHFs or the binding property of Com. $c = \text{Com}(h(M)) = \text{Com}(h(V^*))$ and $M(c) = V^*(c) = r$.

2.2 Canetti et al.'s protocol [23]

KP-PRS rewinding strategy. Before we describe Canetti et al.'s protocol, we first recall the key approach in achieving black-box concurrent zero-knowledge [15, 17, 19, 29]. In the context of the black-box CZK protocol, to increase the chances of extracting the trapdoor, the protocols are designed by using many sequential slots in stage one. Then a simulator using a special rewinding strategy, i.e., the KP-PRS rewinding strategy, guarantees the execution of simulation in polynomial time.

Roughly speaking, assume that a verifier concurrently runs polynomial copies of the CZK protocol, then the total number of messages in the entire execution is bounded. Without losing generality, assume that the total number is b^d, and let the simulator recursively divides the entire transcript into blocks with the smallest size of b. Now the size of each block is b^i $(i \in [d])$, where d is the maximum nesting depth. The simulator then executes in an oblivious method. Specifically, when a new block is completed, it rewinds the verifier at the end point of this block and is oblivious of the messages generated in this block. This strategy demonstrates the following good properties based on their analysis. If the total number of slots is more than $w(b \log_b n)$, the simulator can successfully extract the trapdoor of each session and the maximum nesting depth is at most $O(\log_b n)$. Therefore, when we set $b = w(\log n)$, the simulator can finish in polynomial time.

Canetti's public-coin CZK. By comparing the non-black-box simulation with black-box simulation, Canetti et al. [23] proposed that the process of committing a program M in a slot and mimicking the execution for this slot in a non-black-box simulation can be analogous to the process of committing a slot

and rewinding the execution of that slot in black-box simulation. The only difference is that the non-black-box simulator executes in a straight-line method and it only computes a universal argument (UA) proof about the execution of M. In addition, both the exponential time blowup problem in non-black-box CZK and the recursive nesting problem in black-box CZK are caused by the simulator's recursive run its code. Inspired by the above observation, they proposed a KP-PRS style proving method. Specifically, the simulator divides the messages into blocks of length b^i. Then at the end point of each block, the simulator computes a UA proof (called block-proof) arguing that there exists a code description committed in c such that the program can simulate the execution of this block and outputs a transcript τ. To prove that each slot (c, r) is included in this block, they used a second UA proof (called session-proof) arguing that τ contains the challenge string r. The soundness can be proved as follows, if the block-proof and the corresponding session-proof are both accepted, it means that there exists a program committed in c that can output some messages τ containing r. That is, the program can output the challenge string r. Therefore, the soundness can be obtained by reducing the security to the soundness of the previous Barak's protocol [21].

In Canetti et al.'s protocol [23], the UA proof constructed by the simulator above will not be used immediately. That is, only when a session enters the UA proof stage, the simulator may choose it as a witness. To address this issue, they used a special UA proof, i.e., the offline/online UA proof, which allows the simulator to store all the UA proofs offline (which contains a probabilistically checkable proof (PCP) σ and its Merkle hash tree). Therefore, the simulator needs to remember all previously generated proofs and each commitment of the slot also need to contain all previously generated offline proofs. To avoid storing such large amounts of offline proofs in each commitment, Canetti et al. [23] used a hash-inverting oracle and a common hash function h that can be used by all parties. Specifically, in each slot, the simulator will use c to commit to a code M and the roots of a Merkle hash tree of previously offline UA proofs. When enter the second stage, the simulator needs to prove that there exists some slot (c_i, r_i) such that c_i commits to a program M. Given oracle access to the hash-inverting oracle, the simulator can generate a transcript τ that contains the verifier's challenge string r_i. The oracle contains all previously offline proofs and only responds to the PCP query if the root of the corresponding hash tree of σ is contained in the committed values of the commitment c. This constraint condition guarantees that all the query and answer pairs are computationally determined by the committed values and are independent of the verifier's challenge r. If a cheating prover can generate two accepted oracles but their answers are different from the same PCP query position, then the assumption that the hash function h is collision-resistant will be violated.

2.3 Our techniques

We construct a new $O(\log^{1+\varepsilon} n)$-round protocol by modifying the protocol of [23] in the following way. First, recall that when the simulator enters the point of the end of the block, it needs to construct a block-proof for the program M that mimics the messages of this block. To achieve this goal, when c is generated at the beginning of a slot, c must commit to this M. However at this time, it is not yet known when the corresponding r will appear (it is determined by the verifier). This means that the simulator does not know which block will be the minimal block and contains the pair (c, r). For this reason, Ref. [23] modified each slot to include n commitments in such a case $((c_{i_1}, \ldots, c_{i_n}), r_i)$. Upon analysis, at this point there are at most n blocks that have been opened because the nesting depth of this protocol is at most $O(\log n / \log\log n)$. Now the simulator can commit to all the program states by using $(c_{i_1}, \ldots, c_{i_n})$ at the beginning of each currently open block. However in such a way, the protocol's communication complexity in each slot becomes n times of the traditional Barak's protocol described above. Our approach to this problem is in the spirit of the hash-inverting oracle idea. More specifically, we use a variant of the oracle to allow it to not only respond to the PCP query in a deterministic method but also can respond to the previous state of the simulator in a deterministic method. In particular, when the simulator enters a new slot, it must have stored all the previous commitment values (i. e., the program M) by itself, because it need to use these values as witnesses for the offline PCP proof constructions at each end of block. This

means that all previous states have been deterministic at this time. We can now modify each slot to commit the current program M of the simulator as before, including all the hash values of these previous commitment values stored by the simulator. The intuition is that for a collision resistant hash function, no computationally bounded machine can produce two programs M and M' which are not same but their hash values are equal. Otherwise, we can use this machine to break the security of collision-resistant hash functions. Additional details of the soundness proof are given in Subsection 4.3.

Second, we observed that although we can view the method of construction of the offline PCP proof at each block as the KP-PRS oblivious rewind strategy in black-box simulation, there is a slight difference between the two. In particular, for the black-box simulation in private-coin protocol, the simulator does not know which slot in the main thread will become the matched convincing slot with some auxiliary thread. To successfully extract the trapdoor, the simulator must honestly rewind the whole block in each recursive step. Even many messages ahead in the block do not belong to the preamble part. However, for the public-coin protocol, we used a straight-line non-black-box simulation approach, hence we can avoid such a problem. More specifically, when enter the end of each block and when the construction of the offline PCP proof begins, we only need to simulate a part of the block from the first "unsolved" slot in this block and use its commitment value as the trapdoor. We provide a full description of our simulator for the concurrent zero-knowledge and then provide proof of the soundness and argument of knowledge. More details are given in Subsections 4.1 and 4.2.

Finally, we use the transformation of [30] to construct resettably-sound protocol [4, 31, 32]. Roughly speaking, for each message from the verifier in our original protocol, we replace the randomness needed by the verifier by applying a pseudo-random function f_s to the messages that the verifier has received so far. In such a case, we can obtain a new protocol which is resettably-sound CZK argument of knowledge. Furthermore, we know that the theorem in [32] based on [33] demonstrated that by assuming the existence of one-way functions, every ℓ-round resettably-sound CZK argument of knowledge can be transformed to a $O(\ell)$-round simultaneous resettable zero-knowledge argument of knowledge [4, 32, 34]. Therefore, we provide a new improved construction of the simultaneous resettable ZK argument system with $O(\log^{1+\epsilon}(n))$ round complexity over the corresponding result by [25, 33].

3 Preliminary

3.1 Statistically binding commitment schemes

A commitment scheme consists of two phases, the commit phase and reveal phase. In the first phase, the sender commits to a value and sends the commitment to the receiver. If the commitment keeps the value hidden from the receiver, then this property is referred to hiding. In the second phase, the sender sends the value and the randomness used in the commit phase to the receiver. If the sender can only generate one accepted value and randomness for the receiver, then this property is referred to binding. We say a commitment scheme is statistically binding if the binding property holds against unbounded adversaries, which can be constructed based on the one-way functions assumption [35, 36].

3.2 Witness indistinguishability

Definition 1 (Witness indistinguishability [20]). We say that an interactive protocol (P, V) is witness indistinguishable for the witness relation R_L if for every probabilistic polynomial time (PPT) adversarial verifier V^* and for every two witness sequences $\{w_x^1\}_{x\in L}$ and $\{w_x^2\}_{x\in L}$ such that $(w_x^1, x) \in R_L$ and $(w_x^2, x) \in R_L$, the following ensembles are computationally indistinguishable:

- $\{\text{view}_{V^*}\langle P(x, w_x^1) \leftrightarrow V^*(x)\rangle\}_{x\in L}$,
- $\{\text{view}_{V^*}\langle P(x, w_x^2) \leftrightarrow V^*(x)\rangle\}_{x\in L}$.

In this paper, our WIAOK protocol for language in NP can be obtained by instantiating Blum's protocol [37] by using a statistically binding commitment scheme.

3.3 Concurrent zero-knowledge

Definition 2 (Concurrent zero-knowledge [2]). Let (P, V) be an interactive proof protocol for $L \in \text{NP}$. We say that a verifier V^* is a concurrent adversarial verifier if for any polynomial m, V^* can run m independent copies of the protocol with P and schedule these copies in an arbitrary way. Denote $\text{view}_{V^*}(P(w), V(z))(x)$ as the view of an adversarial verifier $V^*(z)$ when interacting with the prover $P(w)$ on common input x and auxiliary input $w \in R_L(x)$ and z. We say (P, V) is concurrent zero-knowledge if for every concurrent adversarial verifier V^*, there exists a PPT simulator S such that the two ensembles $\{\text{view}_{V^*}(P(w), V(z))(x)\}_{x\in L, z\in\{0,1\}^*}$ and $\{S(x, z)\}_{x\in L, z\in\{0,1\}^*}$ are computationally indistinguishable.

3.4 PCP system

Definition 3 (PCP system [38]). Let (P, V) be an interactive proof system for language L. In a PCP system (P, V), the prover P constructs a redundant proof and allows the PPT verifier to query any bits positions of this proof in an oracle method. More specifically, it satisfies the following:

Completeness. For every $x \in L$ there exists a redundant proof π_x such that V oracle access to π_x will always accept $x \in L$.

Soundness. For every $x \notin L$ and every proof π, V oracle access to π will reject $x \in L$ with probability at least $1 - \text{negl}(|x|)$.

3.5 Universal argument

Definition 4 (Universal argument [38]). Let $T(n) = n^{\log\log n}$, we define a language $L_{\mathcal{U}}$ with witness relation $R_{\mathcal{U}}$ as follows: we say $y = (M, x, t) \in L_{\mathcal{U}}$ iff $M(x) = 1$ such that M is the description of a non-deterministic Turing machine and can decide x within $t < T(|x|)$ steps and define $R_{\mathcal{U}}(y) \stackrel{\text{def}}{=} \{w : (y, w) \in R_{\mathcal{U}}\}$. Based on this definition, every language $L \in \text{NP}$ can be linear time reducible to $L_{\mathcal{U}}$. We say that an interactive protocol (P, V) is a UA system for $L_{\mathcal{U}}$ if it satisfies the following:

Efficient verification. For any $y = (M, x, t)$, V can verify y within a fixed polynomial $p(|y|)$ steps.

Completeness. For every $(y, w) \in R_{\mathcal{U}}$, it holds that $\Pr[(P(w), V)(y) = 1] = 1$. The time spent by $P(y, w)$ is bounded by $p(|M| + T_M(x, w)) \leqslant p(|M| + t)$, where p is a fixed polynomial and $T_M(x, w)$ is the time spent by $M(x, w)$.

Soundness. For every PPT P^*, and every $y \notin L_{\mathcal{U}}$, it holds that $\Pr[(P^*, V)(y) = 1] = \text{negl}(n)$.

Weak proof of knowledge. For every PPT P^*, every $y \in \{0, 1\}^{\text{poly}(n)}$ and every polynomial p, if $\Pr[(P^*, V)(y) = 1] > 1/p(n)$, then there exists a PPT extractor E and a polynomial q such that

$$\Pr_r[\exists w = (w_1, \ldots, w_t) \in R_{\mathcal{U}}(y), \text{ s.t. } \forall i \in [t],\ E_r^{P^*}(1^n, y, i) = w_i] > \frac{1}{q(n)},$$

where the knowledge extractor E with a fixed random tape r is a machine that can oracle access to P^* and output the i-th bit of the witness w. In particular, assume $|w| \leqslant t$, then the extractor E can extract the entire witness w within $\text{poly}(n)\cdot t$ steps, and we refer to this property as the global proof-of-knowledge.

As we explained in Section 2, we will use the special UA defined in [23] in which the construction consists of an offline stage and an online stage. Assume that the common input is x and the private input to prover is $w \in R_L(x)$. We provided a concrete description in Figure 1.

3.6 Forward-secure PRG

Definition 5 (Forward-secure pseudorandom generator [24,39]). We say a function is a forward secure pseudorandom generator (fsPRG) if for every $n, \ell \in N$, $\text{fsPRG}(s, \ell) = (s_\ell, s_{\ell-1}, \ldots, s_1), (\rho_\ell, \rho_{\ell-1}, \ldots, \rho_1)$ and the following hold:

(1) If $\text{fsPRG}(s, \ell) = ((s_\ell, s_{\ell-1}, \ldots, s_1), (\rho_\ell, \rho_{\ell-1}, \ldots, \rho_1))$, then $\text{fsPRG}(s_\ell, \ell-1) = ((s_{\ell-1}, \ldots, s_1), (\rho_{\ell-1}, \ldots, \rho_1))$.

(2) For every polynomial p, the following

- $\{s \leftarrow U_n, (\vec{s}, \vec{\rho}) \leftarrow \text{fsPRG}(s, \ell) : s_t, \vec{\rho}_{\leqslant t}\}_{\ell\in[p(n)], t\in[\ell], n\in N}$,

Online stage:

Step 1: V chooses $h \xleftarrow{R} \mathcal{H}_n$ and sends h to P.

Offline stage:

The Prover P invokes the PCP system to construct a proof σ, and then generates the Merkle hash tree of σ under the hash function h, denote HT and δ are the hash tree and its root, and denote $\pi = \sigma||\text{HT}$ as the offline PCP proof.

Online stage:

Step 2: P sends δ to V.

Step 3: V responds a challenge message r to P.

Step 4: P invokes PCP on (x, r) to construct the PCP queries Q; then sends $\{q, \sigma_q, \text{auth}_q(\sigma)\}_{q\in Q}$ to V, where σ_q is the q-th bit of σ and $\text{auth}_q(\sigma)$ is the certificate path of HT for σ_q.

Verification stage:

V accepts if all the $\{\text{auth}_q(\sigma)\}_{q\in Q}$ are accepted and the $(x, r, \{\sigma_q\}_{q\in Q})$ is accepted by the PCP system.

Figure 1 Special universal argument.

- $\{s_t \leftarrow U_n, \vec{\rho} \leftarrow (U_n)^\ell : s_t, \vec{\rho}_{\leqslant t}\}_{\ell\in[p(n)],t\in[\ell],n\in N}$

are computationally indistinguishable, where $\vec{\rho}_{\leqslant t} = (\rho_t, \rho_{t-1}, \ldots, \rho_1)$, and $U_n \xleftarrow{R} \{0,1\}^n$.

The fsPRG function can be constructed by using PRG which can be based on the one way functions (OWFs) assumption.

4 Public-coin concurrent zero-knowledge

In this section, we give our construction of the public-coin concurrent zero-knowledge protocol and prove Theorem 1. We use the following building blocks:

- 2-round public-coin statistically binding commitment scheme: Com
- 4-round public-coin special-soundness witness indistinguishable proofs: WIAOK
- 4-round public-coin special purpose universal argument: UA

Now consider a language $L \in$ NP, and let the prover and verifier receive a common input $x \in \{0,1\}^n$, $h \in \mathcal{H}$. The auxiliary input to the prover is an NP witness w such that $R_L(x, w) = 1$. Our protocol then consists of three stages as follows:

In stage 1, the prover and the verifier run $k = O(\log n/\log\log n)$ rounds to generate $\{(c_i, r_i)\}_{i\in[k]}$, where r_i is the challenge string with length $4n$, and then P interacts with V in stage 2 to generate the encrypted UA by using the special purpose universal argument. In other words, the honest prover sends messages in a commitment scheme by committing to a dummy value 0^n. In stage 3, the prover runs WIAOK to prove that there exists a valid witness w such that $w \in R_L(x)$ or the opening of the encrypted UA in stage 2 is an accepted block proof for a block transcript τ which contains a challenge string r_i for some $i \in [k]$.

The formal description of public-coin NMZK protocol is described in Figures 2 and 3.

The completeness directly follows from the construction in Figure 3. Below, we give proofs of the concurrent zero knowledge and the soundness.

4.1 Concurrent zero-knowledge

To simplify the exposition, we assume that the verifier V^* is a deterministic Turing machine with a non-uniform advice. Denote N as the total number of messages throughout the execution, and $b = O(\log n)$ is the split factor of the "KP-PRS" strategy. Without loss of generality, we let N be a power of b, i.e., $N = b^d$, where d is bounded by $O(\log n/\log\log n)$. We generate the random-tape by invoking

Oracle $\mathcal{O}$:

$$\mathcal{O}(\delta, q_i) = \begin{cases} (\sigma_{q_i}, \mathrm{auth}_{q_i}(\sigma)), & \text{if } \delta \text{ is the root value of the Merkle tree HT of } \sigma \text{ which} \\ & \text{stored in oracle } \mathcal{O} \text{ and } q_i \in [2^{|\sigma|}]; \\ M, & \text{if } \delta \text{ is the hash value of the description of } M \text{ which} \\ & \text{stored in oracle } \mathcal{O} \text{ and } q_i = -1; \\ \perp, & \text{otherwise.} \end{cases}$$

Here σ is the PCP proof and $\mathrm{auth}_{q_i}(\sigma)$ is the authenticator for σ_{q_i}, and M is the description of a program that contains the code of the simulator and the state of the verifier.

Language Λ_1 :
We say $(h, h_M, h_\tau) \in \Lambda_1$, iff there exist $(M, s, \ell, \mathcal{O}, \tau)$ such that
– $M^{\mathcal{O}}(s, \ell) = \tau$, $h_M = h(M)$, $h_\tau = h(\tau)$, (s, ℓ) is the auxiliary input of M.

Language Λ_2 :
We say $(h, h_M, h_\tau, c_k, r_k) \in \Lambda_2$, iff there exist $(M, M_k, \rho_k, y, \mathcal{O}, \tau)$ such that
– $h_M = h(M)$, $M_k^{\mathcal{O}}(y) = M$, y is the auxiliary input M_k such that $|y| = n$,
– $h_\tau = h(\tau)$, $c_k = \mathrm{Com}(h(M_k), \rho_k)$ such that τ contains r_k.

Figure 2 Block-proof and session-proof.

Common input: $x \in L$ $(|x| = n)$.
Auxiliary input to P : $w \in R_L(x)$.
Common hash function : $h \in \mathcal{H}$.

Stage 1
For $k \in [K]$, repeat the following two steps sequentially:
$P \longrightarrow V$:] $c_k = \mathrm{Com}(0^n)$,
$V \longrightarrow P$:] $r_k \leftarrow_R \{0,1\}^{4n}$.

Stage 2
P and V run two special UA for a statement of the same length as $(h, 0^n, 0^n) \in \Lambda_1$ and for a statement of the same length as $(h, 0^n, 0^n, c, r) \in \Lambda_2$.
$P \longrightarrow V$:] $\alpha_1' = \mathrm{Com}(0^n), \alpha_2' = \mathrm{Com}(0^n)$,
$V \longrightarrow P$:] $\beta_1 \leftarrow_R \{0,1\}^n, \beta_2 \leftarrow_R \{0,1\}^n$.

Stage 3
$P \Longleftrightarrow V$: P and V execute the 3-round WIAOK that $\exists$ w s.t., $(x, w) \in R_L$ **or** for some $k \in [K]$, there exists a statement $\mathrm{st}_1 = (h, h_M, h_\tau)$ and a statement $\mathrm{st}_2 = (h, h_M, h_\tau, c_k, r_k)$ such that
1. $\exists(\alpha_1, \gamma_1, \rho_1)$ such that $\alpha_1' = \mathrm{Com}(\alpha_1, \rho_1)$ and $(h, \alpha_1, \beta_1, \gamma_1)$ is an accepting UA proof for $\mathrm{st}_1 \in \Lambda_1$ **and**
2. $(\alpha_2, \gamma_2, \rho_2)$ such that $\alpha_2' = \mathrm{Com}(\alpha_2, \rho_2)$ and $(h, \alpha_2, \beta_2, \gamma_2)$ is an accepting UA proof for $\mathrm{st}_2 \in \Lambda_2$.

Figure 3 Public-coin concurrent zero-knowledge argument of knowledge CZKAOK.

$\mathrm{fsPRG}(s, b^d) = ((s_{b^d}, s_{b^d-1}, \ldots, s_1), (\rho_{b^d}, \rho_{b^d-1}, \ldots, \rho_1))$, where $s \in \{0,1\}^n$ is the random seed and each randomness $\rho_j \in \{0,1\}^n$ will be used to generate the j-th message for prover.

We define the oracle $\mathcal{O}$ as $(\boldsymbol{M}, \boldsymbol{U})$, where $\boldsymbol{M}$ can be used to store the commitment values in stage 1 and $\boldsymbol{U}$ stores the block-proofs and session-proofs generated in stage 2. We use $\boldsymbol{T}$ to store the interactive messages simulated so far. In the beginning, we add the code description of the simulator S and V^* as the first two records to $\boldsymbol{M}$ and initialize $\boldsymbol{U}$ and $\boldsymbol{T}$ to be empty. Next, we give a detailed description of the main program $S(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s, b, d)$.

Main program. For $i \in [d]$, $S(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s_i, b, i)$ attempts to simulate b^i messages of a block of level i, with the help of the oracle $\mathcal{O}$.

(1) If $i = 1$, it means that this block is the minimal size of a block, then execute $M^{\mathcal{O}}(s_i, b)$ to emulate b rounds messages by internally interacting with V^* from the current history $\boldsymbol{T}$, where M is a interactive

Turing machine with the code description of S and V^*, including the current states of them. More details will be given below.

(2) If $i > 1$, then for $j = 1$ to b, do the following:

(i) Recovers all the random seeds $(s_{i-1}^b, s_{i-1}^{b-1}, \ldots, s_{i-1}^1)$ for b sub-blocks of level $i-1$ by invoking $fsPRG(s_i, b^i)$.

(ii) Invokes $S(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s_{i-1}^j, b, i-1)$, this is a recursive process that simulates the j-th sub-block from current history.

(iii) Invokes $S_{ua}(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s_{i-1}^j, b, i-1)$, this procedure is used to construct the block-proof and the session-proofs for slots included in this block but have not been "solved" as yet.

Finally, the simulator $S(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s, b, d)$ outputs all the messages in $\boldsymbol{T}$.

Procedure of program M. The program $M^{\mathcal{O}}(s_i, b)$ attempts to simulate b messages of a base block. First, M invokes $fsPRG(s, b)$ to generate all the random tape, i.e., $(\rho_b, \rho_{b-1}, \ldots, \rho_1)$, then for $i = 1$ to b do the following: Simulate Stage 1. If the next scheduled message is the kth slot message of session i, S provides a commitment c_i^k to $M_i^k((\cdot,\cdot),(\cdot),V^*,(\vec{m},\vec{u},\boldsymbol{T}))$, where M_i^k is an interactive Turing machine that includes the code description of S and V^*, including the current state of them. The first two parameters of M_i^k will be given when M_i^k is used as the witness to construct the UA proofs in stage 2. The third parameter is given when using it to query the program description of M from oracle $\mathcal{O}$, $\vec{u}$ denotes all the roots of the Merkle tree of the PCP proofs (contained block-proofs and session-proofs) stored in $\boldsymbol{U}$, $\vec{m}$ denotes all the hash values of the programs stored in $\boldsymbol{M}$.

Simulate Stage 2. If the next scheduled message is the UA message in Stage 2 in session i, according to our previous analysis, at this point there must exist a block transcript τ and $k \in [K]$ such that the block-proof with respect to τ for statement st_1 and the corresponding session-proof with respect to a slot (c_k, r_k) for statement st_2 have been computed by S_{ua}. Therefore, we just need to retrieve both the roots (δ_1, δ_2) of the Merkle hash tree of these proofs from $\boldsymbol{U}$ to compute the prover messages p in commitment schemes, i.e., (α_1', α_2') and let v be the verifier response.

Simulate Stage 3. If the next scheduled message is the WIAOK message in stage 3, at this point we have computed a part of fake witness, i.e., $(\mathrm{st}_1, \mathrm{st}_2, \alpha_1, \alpha_2)$ in stage 2. The remaining part is (γ_1, γ_2) which includes the responses to the challenge of UA in stage 2, i.e., (β_1, β_2). We can compute this part and the previous step as well. In other words, we first retrieve the Merkle hash tree of the corresponding block-proof and session-proof from $\boldsymbol{U}$ and then respond to the challenge (β_1, β_2) by invoking the universal argument system. We now have all the fake witnesses and compute the prover messages p and let v be the verifier response.

Procedure of S_{ua}. $S_{ua}(1^n, x, V^*, \mathcal{O}, \boldsymbol{T}, s_{i-1}^j, b, i-1)$ attempts to construct block-proof and session-proofs for all the unsolved slots contained in this block. We assume, without losing generality, a transcript $\tau = c||\cdots||r$ to be the minimal size in this block, which contains all the slots where the offline proofs of the corresponding sessions have not been constructed yet. In addition, the total number of messages of τ is ℓ. Denote a set $J = \{(\hat{c}_j, \hat{r}_j)\}_{j\in[m]}$ as these unsolved slots ordered according to the time of the slot opens, where m is polynomial in n, and each pair $(\hat{c}_j, \hat{r}_j)$ corresponds to some slot (c_k, r_k) in some session i. The statement st_1 in language Λ_1 can be denoted as $(h, h_M, h_\tau) \in \Lambda_1$, and each statement st_2 in session i can be written as $(h, h_M, h_\tau, c_k, r_k) \in \Lambda_2$.

- Construct block-proof. We first consider the point at which the simulator generates the first message (i. e., the message c) in τ. Note that from this point onward, the simulator which is given only queries to the oracle is able to continue generating the transcript τ and arrive at the challenge r. Because the commitment values M_1 for c in table $\boldsymbol{M}$ stores all the code of the simulator (include V^*) and the internal state. From the property of the forward secure pseudorandom generator, we can use the random seed s to recover the seed s_τ (which can furthermore generate all the randomness used in this transcript τ) as before. So M_1 must be able to perform the same execution as the simulator with the help of the same oracle. Now we have all the universal argument witnesses: $(M_1, s_\tau, \ell, \mathcal{O}, \tau)$. We can compute the block-proof σ and the corresponding hash tree HT. Finally we extend the table $\boldsymbol{U}$ with a new $\sigma||\mathrm{HT}$ related to the statement st_1 for all sessions related to concerning J.
- Construct session-proof. This step is slightly more complicated than before. More specifically, our

session-proof is quite different from the method given in [23]. In their paper [23], each slot in set J has n commitments $(c_{i,1}, \ldots, c_{i,n})$ corresponding to all the same blocks that have been opened, and a proof is then generated with respect to one of these commitments. However, in the paper, we only have one commitment in each slot, and each commitment value in set J is different. Therefore, it cannot be directly used to generate session-proof as before. But we observe that there exists a polynomial relation between the commitment $\hat{c}_1 = \mathrm{Com}(h(M_1))$ and $\hat{c}_j = \mathrm{Com}(h(M_j), \rho_j),\ j \geqslant 2$. Specifically, w.l.o.g, assume the commitment $\hat{c}_1$ is the y-th round message in the whole interaction, and the program M_j has committed to the hash value δ_1 of M_1 in table $\boldsymbol{M}$. Then, by the definition of our hash inverting oracle in Figure 2, for any $j \in [m]$, it must hold that $M_j^{\mathcal{O}}(y) = M_1$, where M_j queries the oracle $\mathcal{O}$ the yth program M_1 with the root δ_1. Now we can use this relationship to construct each session-proof, more specifically, for each $j \in J$, retrieve the message (M_1, M_j, s, τ) from tables $\boldsymbol{M}$ and $\boldsymbol{T}$, then compute offline PCP proof for statement $(h, h_M, h_\tau, c_j, r_j) \in \Lambda_2$ using witness $(M_1, M_j, \rho_j, y, \mathcal{O}, \tau)$, finally extend the table $\boldsymbol{U}$ with a new $\sigma||\mathrm{HT}$ about statement st_2 for session related to slot (c_j, r_j).

4.2 Indistinguishability of the simulation

Before proving the indistinguishability of the simulation, we first analyze the total time spent by the simulator. Indeed, we can observe that in our new protocol, the expensive part is the same as [23]. In other words, the generation of the block-proof for the block and the recursive regeneration of all session-proofs for other sessions whose stage 2 contained in this block. Therefore, based on Canetti et al.'s analysis in [23], the simulator must be executed in polynomial time.

Next, we use the hybrid argument to demonstrate the indistinguishability of the simulation and consider $2N$ hybrid experiments as follows:

Experiment Hyb^i, $0 < i \leqslant N$. The first i communication rounds are simulated by simulator S with the pseudo-randomness and fake witness; the $j > i$ rounds are simulated by simulator S with true randomness and the true witness w.

Next we define hybrid Hyb_+^i which does the same as the hybrid Hyb^i except that when the simulator simulates the i-th round, it follows the honest prover strategy by using the real witness. Now we observe that the only difference between Hyb_+^i and Hyb^i is the witness used in the WIAOK in stage 3. Therefore, from the hiding property of the Com and the witness indistinguishability property of the WIAOK, we have that of Hyb_+^i and Hyb^i are computationally indistinguishable.

Next, we observe that the only difference between Hyb_+^i and Hyb^{i+1} is the randomness used in the $(i+1)$ round. The former is the true randomness and the latter is the pseudo-randomness. Thus, the indistinguishability of Hyb_+^i and Hyb^{i+1} follows directly from the forward security of the PRG.

Finally, we can observe that the output of Hyb^N is identical to S and the output of Hyb^0 is identical to the real view. Because there are at most polynomial hybrids in this experiment, we can conclude that the output of S is indistinguishable from the output of the real interaction.

4.3 Soundness and argument of knowledge

Note that by the argument of knowledge property of WIAOK in stage 3, we have that for any cheating prover P^* who is able to successfully convince the verifier V to accept this protocol, there must exist an extractor E which given oracle access to P^* can extract a witness w for stage 3. To prove the argument of knowledge property, we can prove that w is a universal argument witness for UA proof transcript in stage 2 only with negligible probability.

Assume for contradiction that there exists a cheating prover P^* and for infinitely many $n \in N$, there exists $x \notin L$ such that the following event happens with a noticeable probability $\epsilon = 1/p$, where p is polynomial in n. When P^* successfully convince the honest verifier V to accept, we can construct an extractor E to recover the transcript of the universal argument for its kth slot with respect to a block transcript τ. We fix this k and denote this event by Event-k.

We refer to the point where the prover just sent a commitment c_k in the kth slot as prefix1, and we refer to the point where the prover just sent the commitments in stage 2 as prefix2. Then, we consider the following experiment Exp with a prover P^{**}.

• P^{**} internally runs P^* and acts as an honest verifier that interacts with P^*. After the execution of the full protocol then P^{**} runs the WIAOK extractor E. If the event Event-k happens, that is E extracts witnesses above, then P^{**} forwards part of witness $(\mathrm{st}_1, \mathrm{st}_2, \alpha_1, \alpha_2)$ to an external verifier and starts the UA system to prove st_1 and st_2 simultaneously. If E fails, then P^{**} aborts.

• P^{**} receives the UA challenge messages (β_1', β_2') from the external UA verifier. At this time P^{**} rewinds P^* to prefix2 and forwards (β_1', β_2') to P^*.

• Finally, P^{**} completes the rest of the protocol as an honest verifier with the prover P^*. If the event Event-k happens again, that is P^{**} has extracted $(\hat{st}_1, \hat{st}_2, \hat{\alpha}_1, \hat{\alpha}_2, \hat{\gamma}_1, \hat{\gamma}_2)$, then P^{**} forwards this message to the external UA verifier, otherwise P^{**} aborts.

Now we analyze the probability that P^{**} completes this entire experiment without aborting and convinces the external UA verifier to accept the proof transcript. We first observe that since the probability of the event Event-k happens at least ϵ. Therefore, from the average argument we have that at least $\frac{\epsilon}{2}$ (over the P^{**} random coins) fraction of prefix1 executions, the event Event-k happens at least $\frac{\epsilon}{2}$. Under this condition, at least $\frac{\epsilon}{4}$ fraction of prefix2 executions, the event Event-k happens at least $\frac{\epsilon}{4}$. Therefore, under these two conditions, the probability that P^{**} successfully extracts both valid witnesses is at least $(\frac{\epsilon}{4})^2$.

Let Incon denote the event that the extracted witnesses in Exp are inconsistent. Specifically, this implies that at least one of the following pairs is not equal: (i) α_1 and $\hat{\alpha}_1$, (ii) α_2 and $\hat{\alpha}_2$, (iii) st_1 and $\hat{st}_1$, (iv) st_2 and $\hat{st}_2$. It is not hard to see that $\Pr[\mathrm{Incon}] \leqslant \mathrm{negl}(n)$. Otherwise, we can break either the statistically-binding property of commitment or the collision-resistant assumption.

Now, we can conclude that the probability P^{**} convinces the external UA verifier to accept that $(h, \hat{\alpha}_1, \hat{\beta}_1, \hat{\gamma}_1)$ is an accepting UA proof for the statement st_1 and $(h, \hat{\alpha}_2, \hat{\beta}_2, \hat{\gamma}_2)$ is an accepting UA proof for the statement st_2 is at least $\frac{\epsilon}{2} \cdot \frac{\epsilon}{2} \cdot (\frac{\epsilon}{4})^2$. Using the definition in Subsection 3.5, there exists a UA extractor E^* that can extract the witness for st_1 and witness for st_2 with probability at least $\mathrm{poly}(\frac{\epsilon}{2} \cdot \frac{\epsilon}{2} \cdot (\frac{\epsilon}{4})^2)$. The first statement implies that there exists a program M such that $h_M = h(M)$ and $h_\tau = h(\tau)$ and on input a short input (s, ℓ), $M^{\mathcal{O}}(s, \ell)$ can generate a partial transcript τ. The second statement implies that for the fixed slot (c_k, r_k), there exists an additional program M_k and an index parameter $y \in \{0,1\}^n$ such that $c_k = \mathrm{Com}(h(M_k))$ and $M_k^{\mathcal{O}}(y)$ can output the description of M. Furthermore, the transcript τ proved in st_1(also in st_2) contains a challenge string r_k. Therefore, we can infer that based on the input of the three parameters $((s, \ell), (y))$, M_k can also predict the random challenge r_k.

Next, we can conclude that if there exists a program P^{**} be able to prove st_1 and st_2 by interaction with the external verifier with probability $\frac{\epsilon}{2} \cdot \frac{\epsilon}{2} \cdot (\frac{\epsilon}{4})^2$, then there exists a program $n^{\log\log n}$-time E* can extract a program M_k predict r_k with probability at least $\mathrm{poly}(\frac{\epsilon}{2} \cdot \frac{\epsilon}{2} \cdot (\frac{\epsilon}{4})^2)$-$\mathrm{negl}(n)$, denote this probability as ϵ_1.

Next, P^{**} rewinds to the point prefix1 and completes the experiment EXP honestly with new random string r_k'. Then, we can extract a program M_k' that predicts r_k' with a probability of at least $\mathrm{poly}(\frac{\epsilon}{4})^3$ again, denote this probability as ϵ_2.

Recall that $c_k = \mathrm{Com}(h(M_k)) = \mathrm{Com}(h(M_k))$, we first analyze the probability of an arbitrary program M such that $c = \mathrm{Com}(h(M))$ can output the verifier's challenge string r. As described before, $\mathcal{O}$ will only answer M's query if M contains the root of the Merkle tree of the corresponding PCP proof or the hash value of a description of an arbitrary program M'. Based on the nice property of the Merkle tree from [38], if a cheating prover can generate two accepted oracles but their answers are different to the same PCP query position, then the assumption that the hash function h is collision-resistant will be violated. Here we say that two accepted oracles that answer differently means that they contain two different PCP proofs (respectively, two different description of program) but their roots of the Merkle trees of these two PCP proofs (respectively, their hash values) are the same. This also means that all the information that program M learns does not depend on the verifier's challenge r, and they are all

P and V on common input x, h, and private input w for P:

1. At the beginning, V chooses a random seed s for a pseudorandom function $f_s : \{0,1\}^* \to \{0,1\}^{\mathrm{poly}(n)}$ P and V run protocol CZKAOK with the following modifications;
2. For each round from V to P, V first computes its randomness by applying f_s on the message it receives so far, and then it invokes the protocol CZKAOK with this new randomness to generate its response in this round.

Figure 4 Resettably-sound concurrent zero-knowledge argument of knowledge (rsCZKAOK).

computationally determined by the original state of M, which is committed in c and the short input $((s,\ell),(y))$.

Therefore, for a single input $(s,\ell) \in \{0,1\}^{2n}$ and $y \in \{0,1\}^n$, there are a total of 2^{3n} possible outputs of M. However, the challenge $r \in \{0,1\}^{4n}$, and so the probability that an arbitrary program M can predict a random r is at most 2^{-n}. According to our analysis, because $c_k = \mathrm{Com}(h(M_k)) = \mathrm{Com}(h(M'_k))$, from the statistically binding property of Com, we have that $h(M_k) = h(M'_k)$ except with negligible probability. We know that the program M_k can predict a fresh random r'_k is negligible (this is because $\Pr[r_k = r'_k] = 2^{-4n}$). Therefore we infer that $M_k \neq M'_k$ except with negligible probability. However, because $h(M_k) = h(M'_k)$, we have obtained a collision in the hash function h with probability of at least $\epsilon_1 \cdot \epsilon_2$ (=1/poly(n)), thereby we break the security of the CRHFs.

Therefore, we can conclude that for every k, the probability that the event Event-k occurs is negligible, and the extractor E^* must output a witness for $x \in L$.

5 Towards simultaneous resettable zero-knowledge argument

In this section, we apply the transformation of [30] to the public-coin CZK argument of knowledge in Figure 3 to achieve resettably-sound CZK argument of knowledge. In particular, we modify the verifier V that samples a seed s for a pseudorandom function (PRF) f_s at the beginning of the protocol and then generate each verifier message by applying f_s to the current history message it received. We provide the transformation in Figure 4.

We only provide a lemma concerning Figure 4, which can be immediately obtained from [30], and we ignore the proof of this part.

Lemma 1. Assume the existence of OWFs, the protocol rsCZKAOK in Figure 4 is a resettably-sound concurrent zero-knowledge argument of knowledge system.

Next, we rely on the following lemma from [32], which is based on [33].

Lemma 2. Assume the existence of OWFs, there exists a transformation that takes any ℓ-round resettably-sound CZKAOK for NP and outputs a $O(\ell)$-round simultaneously resettable ZKAOK protocol for the same language.

Therefore, we provide a new construction of a simultaneous resettable ZKAOK system with an improvement round complexity over the corresponding result by [25, 33].

6 Conclusion

In this paper, we presented an alternative construction of fully public-coin concurrent zero knowledge based on the existence of the collision resistance hash functions, and provided an explicit zero-knowledge simulator description and detailed proofs. We also given a variant of the straight-line simulator based on Barak's non-black box simulation technology, which may be of independent interest. Based on the results, we can obtain a new construction of a simultaneous resettable ZK argument system.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant No. 61772521), Key Research Program of Frontier Sciences, Chinese Academy of Sciences (Grant No. QYZDB-SSW-SYS035),

and Open Project Program of the State Key Laboratory of Cryptology.

References

1 Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, Providence, 1989. 186–208
2 Dwork C, Naor M, Sahai A. Concurrent zero-knowledge. In: Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, Dallas, 1998. 409–418
3 Chung K M, Ostrovsky R, Pass R, et al. 4-round resettably-sound zero knowledge. In: Proceedings of the 11th International Conference on Theory of Cryptography (TCC), San Diego, 2014. 192–216
4 Chongchitmate W, Ostrovsky R, Visconti I. Resettably-sound resettable zero knowledge in constant rounds. In: Proceedings of the 15th International Conference on Theory of Cryptography (TCC), Baltimore, 2017. 111–138
5 Chung K M, Pass R, Seth K. Non-black-box simulation from one-way functions and applications to resettable security. SIAM J Comput, 2016, 45: 415–458
6 Ostrovsky R, Scafuro A, Venkitasubramaniam M. Resettably sound zero-knowledge arguments from OWFs — The (semi) black-box way. In: Proceedings of the 12th International Conference on Theory of Cryptography (TCC), Warsaw, 2015. 345–374
7 Benhamouda F, Lin H. k-round MPC from k-round OT via garbled interactive circuits. In: Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, 2018. 500–532
8 Garg S, Srinivasan A. Two-round multiparty secure computation from minimal assumptions. In: Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, 2018. 468–499
9 Ishai Y, Mittal M, Ostrovsky R. On the message complexity of secure multiparty computation. In: Proceedings of the 21st IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC), Rio de Janeiro, 2018. 698–711
10 Badrinarayanan S, Goyal V, Jain A, et al. Round optimal concurrent MPC via strong simulation. In: Proceedings of the 15th International Conference (TCC), 2017. 743–775
11 Garg S, Kiyoshima S, Pandey O. A new approach to black-box concurrent secure computation. In: Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, 2018. 566–599
12 Broadnax B, Döttling N, Hartung G, et al. Concurrently composable security with shielded super-polynomial simulators. In: Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, 2017. 351–381
13 Badrinarayanan S, Khurana D, Ostrovsky R, et al. Unconditional UC-secure computation with (stronger-malicious) PUFs. In: Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, 2017. 382–411
14 Kiyoshima S, Lin H, Venkitasubramaniam M. A unified approach to constructing black-box UC protocols in trusted setup models. In: Proceedings of the 15th International Conference (TCC), Baltimore, 2017. 776–809
15 Richardson R, Kilian J. On the concurrent composition of zero-knowledge proofs. In: Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Prague, 1999. 415–431
16 Canetti R, Kilian J, Petrank E, et al. Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: Proceedings of Annual ACM Symposium on Theory of Computing (STOC), Heraklion, 2001. 570–579
17 Prabhakaran M, Rosen A, Sahai A. Concurrent zero knowledge with logarithmic round-complexity. In: Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS), Vancouver, 2002. 366–375
18 Goldreich O, Kahan A. How to construct constant-round zero-knowledge proof systems for NP. J Cryptol, 1996, 9: 167–190
19 Pass R, Dustin Tseng W L, Venkitasubramaniam M. Concurrent zero knowledge, revisited. J Cryptol, 2014, 27: 45–66
20 Feige U, Shamir A. Witness indistinguishable and witness hiding protocols. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC), Baltimore, 1990. 416–426
21 Barak B. How to go beyond the black-box simulation barrier. In: Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS), 2001. 106–115
22 Pass R, Rosen A, Tseng W L D. Public-coin parallel zero-knowledge for NP. J Cryptol, 2013, 26: 1–10
23 Canetti R, Lin H, Paneth O. Public-coin concurrent zero-knowledge in the global hash model. In: Proceedings of the 10th Theory of Cryptography Conference, Tokyo, 2013. 80–99
24 Chung K, Lin H, Pass R. Constant-round concurrent zero knowledge from p-certificates. In: Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, 2013. 50–59
25 Goyal V. Non-black-box simulation in the fully concurrent setting. In: Proceedings of Symposium on Theory of Computing Conference (STOC), Palo Alto, 2013. 221–230
26 Kiyoshima S. An alternative approach to non-black-box simulation in fully concurrent setting. In: Proceedings of the 12th Theory of Cryptography Conference (TCC), Warsaw, 2015. 290–318
27 Pandey O, Prabhakaran M, Sahai A. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In: Proceedings of the 12th Theory of Cryptography Conference (TCC), Warsaw, 2015. 638–667
28 Chung K, Lin H, Pass R. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In: Proceedings of the 35th Annual Cryptology Conference (CRYPTO), Santa Barbara, 2015. 287–307

29 Kilian J, Petrank E. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In: Proceedings of Annual ACM Symposium on Theory of Computing, Heraklion, 2001. 560–569

30 Barak B, Goldreich O, Sha G, et al. Resettably-sound zero-knowledge and its applications. In: Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS), 2001. 116–125

31 Bitansky N, Paneth O. On non-black-box simulation and the impossibility of approximate obfuscation. SIAM J Comput, 2015, 44: 1325–1383

32 Chung K M, Ostrovsky R, Pass R, et al. Simultaneous resettability from one-way functions. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, 2013. 60–69

33 Deng Y, Goyal V, Sahai A. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Atlanta, 2009. 251–260

34 Cho C, Ostrovsky R, Scafuro A, et al. Simultaneously resettable arguments of knowledge. In: Proceedings of the 12th International Conference on Theory of Cryptography (TCC), Warsaw, 2015. 530–547

35 HÅstad J, Impagliazzo R, Levin L A, et al. A pseudorandom generator from any one-way function. SIAM J Comput, 1999, 28: 1364–1396

36 Naor M. Bit commitment using pseudorandomness. J Cryptol, 1991, 4: 151–158

37 Blum M. How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, 1986. 1444–1451

38 Barak B, Goldreich O. Universal arguments and their applications. SIAM J Comput, 2008, 38: 1661–1694

39 Bellare M, Yee B. Forward-security in private-key cryptography. In: Proceedings of The Cryptographers' Track at the RSA Conference, San Francisco, 2003. 1–18