Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 48 (2005)

Concurrently Knowledge-Extractable Resettable-ZK in the Bare Public-Key Model

Moti Yung* Yunlei Zhao†

Abstract

We present constant-round *concurrently knowledge-extractable* black-box resettable zero-knowledge (rZK-CKE) arguments for $\mathcal{NP}$ in the bare public-key (BPK) model. We give minimal (sub-exponential) hardness assumption based protocols as well as round-optimal protocols (still under general hardness assumptions). To our knowledge, our protocols are the first ZK protocols that provably provide both resettable/concurrent prover security and concurrent verifier security in public-key models. Here, the notion of *concurrent knowledge-extractability* roughly means that no malicious polynomial-time prover can convince an honest verifier of any (whether false or true) statement without "knowing" a corresponding $\mathcal{NP}$-witness even by concurrently interleaving interactions in public-key models when verifiers register public-keys. We show that concurrent knowledge-extractability is *strictly* stronger than "concurrent soundness" (under any sub-exponentially strong one-way permutation) for concurrently proving $\mathcal{NP}$ statements to honest verifiers with public-keys. In particular, we show that previous concurrent zero-knowledge protocols in the BPK model that achieved concurrent soundness security and are also traditional arguments of knowledge actually fail to satisfy this stronger security notion (this is demonstrated by concrete attacks). Our work deepens the understanding of the subtleties of concurrent verifier security in public-key settings, and may serve as building blocks in designing other round-efficient concurrently secure protocols in public-key models that use, in particular, argument of knowledge (AOK) protocols as building blocks.

*RSA Laboratories and Department of Computer Science, Columbia University, New York, NY, USA. moti@cs.columbia.edu

†Software School, School of Information Science and Engineering, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn

ISSN 1433-8092

1 Introduction

Resettable zero-knowledge (rZK). rZK is the strongest version of the remarkable notion of zero-knowledge (ZK) [35] to date. It was put forth by Canetti, Goldreich, Goldwasser and Micali [11], motivated by implementing zero-knowledge provers using smart-cards or other devices that may be (maliciously) reset to their initial conditions and/or cannot afford to generate fresh randomness for each new invocation. rZK also preserves the prover's security when the protocol is executed concurrently in an asynchronous network like the Internet (in fact, rZK is a generalization and strengthening of the notion of concurrent zero-knowledge (cZK) introduced by Dwork, Naor and Sahai [22]).

The bare public-key model. To get constant-round rZK protocols, [11] introduced a simple model with very appealing trust requirement, the *bare public-key* (BPK) model. 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. (The model does allow dynamic key registrations and readers are referred to [11] for the details of dealing with dynamic key registrations.) In particular, an adversary may deposit polynomially many (possibly invalid or fake) public keys without any guarantee on the properties of the registered public-keys. The BPK model is thus very simple, and it is in fact a weaker version of the frequently used public-key infrastructure (PKI) model, which underlies any public-key cryptosystem or any digital signature scheme. More discussions on the BPK model are presented in Appendix A.

Verifier security in public-key models. Verifier security in public-key models when verifiers register public-keys turns out to be more complicated and subtle than otherwise. Micali and Reyzin showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger: one-time, sequential, concurrent and resettable soundness [44]. In this work, we focus on concurrent soundness, which roughly means that a malicious prover P^* cannot convince the honest verifier V of a *false* statement even when P^* is allowed multiple interleaving interactions with V. Micali and Reyzin also showed that any (resettable or not) black-box ZK protocols with concurrent soundness in the BPK model (for non-trivial languages outside $\mathcal{BPP}$) must run at least four rounds [44]. It is also shown in [3, 44] that (whether resettable or not) *black-box* ZK *arguments* with resettable soundness only exist for trivial (i.e, $\mathcal{BPP}$) languages (whether in the BPK model or not). Thus, it is commonly suggested that concurrent soundness might be the best one can achieve for (concurrent) verifier security of black-box (resettable) ZK arguments in the BPK model [18].

1.1 Our contributions

- **Result:** We show that both *concurrent soundness* and *traditional argument of knowledge* do not guarantee concurrent verifier security in public-key models, by presenting concrete attacks to natural existing concurrent zero-knowledge protocols in the BPK model that achieved concurrent soundness and are also normal arguments of knowledge. This motivates us to introduce a stronger security notion for concurrently proving $\mathcal{NP}$ statements to honest verifiers with registered public-keys, named "(same public-key sub-exponential-time) *concurrent knowledge-extractability*", along with motivations and justifications. Here, concurrent knowledge-extractability roughly means that no malicious polynomial-time prover can convince an honest verifier of any (whether false or true) statement without "knowing" a corresponding $\mathcal{NP}$-witness even by concurrent interleaving interactions with the honest verifier in public-key models when verifiers register public-keys. We show that contrary to the above mentioned belief, concurrent knowledge-extractability is strictly stronger than concurrent soundness (under any sub-exponentially strong OWP) for concurrent verifier security in public-key models. Actually, as we shall see, *no* previous rZK or cZK protocols in the BPK model could be provably secure under this stronger concurrent verifier security. That is, ZK protocols that provably provide both concurrent/resettable prover security and concurrent verifier security in public-key models are unknown previously. (We also show in Section 3 why it is so hard to achieve concurrent knowledge-extractability for ZK protocols in the BPK model.) This deepens our understandings of the subtleties of concurrent verifier security in public-key models.

In comparison with the notions of "non-black-box argument of knowledge" of [3] and "concurrent

argument of knowledge" of Di Crescenzo and Visconti [20], the knowledge-extraction in our notion is done in *super-polynomial-time* with *black-box* accessing the concurrent malicious prover and using the *same* verifier's public-key. We remark that super-polynomial-time black-box knowledge-extraction is intrinsic in the resettable setting, as resettable (ZK or WI) protocols with black-box polynomial-time knowledge-extraction are possible only for trivial languages [11, 3]. Also, as we shall see, same public-key knowledge-extraction with complexity leveraging (allowed by the sub-exponential assumptions) appears to be the only way at present to bypass the obstacle of concurrent general composition [43, 37] for achieving concurrently knowledge-extractable ZK protocols in the BPK model. We also note that, to our knowledge, it is unknown whether traditional proof/argument of knowledge is strictly stronger than traditional soundness (in the plain model or in public-key models). There are many (zero-knowledge) protocols (e.g., the protocols of [34, 32]) that are sound but may not *provably* be argument/proof of knowledge. But, for those protocols we do not know how a malicious prover can indeed convince a true statement without knowing any witness.

Then, we show how to achieve constant-round concurrently knowledge-extractably secure rZK (rZK-CKE) arguments for $\mathcal{NP}$ in the BPK model, which leads us to new tools (that might be possibly independent interest) and to answering some basic open questions (since rZK-CKE implies concurrently sound rZK and since there, questions like reduced complexity assumptions were open). More specifically we have the following results:

- **(Main) Result:** We achieve constant-round rZK-CKE arguments for $\mathcal{NP}$ in the BPK model under the minimal (sub-exponential) hardness assumptions. Specifically, we present 7-round (resp. 6-round) rZK-CKE arguments for $\mathcal{NP}$ in the BPK model under any (sub-exponentially strong) one-way function OWF (resp. one-way permutation OWP). To this end, we present constant-round resettable witness indistinguishability (rWI) *arguments* for $\mathcal{NP}$ *in the standard model* under minimal (sub-exponential) hardness assumptions, which might be possibly of independent interest.

- **Result:** We achieve *round-optimal* (i.e., 4-round) rZK-CKE arguments for $\mathcal{NP}$ in the BPK model under any (sub-exponentially strong) OWP and any (standard polynomially secure) preimage-verifiable OWF. Note that preimage-verifiable OWF is a generic and actually quite weak hardness assumption that includes, in particular, any certified one-way permutation and any 1-1 length-preserving one-way function. *This implies that round-optimal rZK-CKE arguments for* $\mathcal{NP}$ *in the BPK model can be based on any certified one-way permutation.* To this end, we develop a one-round trapdoor commitment scheme (that is based on any OWP), which might be possibly of independent interest.

- **Result:** We also present *round-optimal* rZK-CKE arguments for $\mathcal{NP}$ in the BPK model under any (sub-exponentially strong) OWP and the existence of (standard polynomially secure) zaps. Note that the existence of (single-theorem) NIZK proofs for $\mathcal{NP}$ implies the existence of zaps [23].

Since zero-knowledge plays a central role in cryptographic protocol design and rZK is a generalization of the notion of cZK, and also since the BPK model is a very fundamental cryptographic model, we suggest that our work could be potentially relevant to public-key cryptography in a potentially large scope. In particular, our protocols could be potentially employed as building blocks to achieve other round-efficient concurrently secure schemes in public-key models that use, in particular, AOK protocols as building blocks. Note also that (zero-knowledge) argument of knowledge is itself a very basic cryptographic primitive.

Details of related works and comparisons are presented in Appendix B, due to space limitation.

2 Preliminaries

Definitions of concurrent soundness, rZK and rWI in the BPK model. We assume the readers are familiar with these definitions, presented in detail in Appendix C.

Major cryptographic tools used. We also assume that readers are familiar with the following cryptographic tools whose detailed descriptions are in Appendix D.

(1) Preimage-verifiable one-way functions: A OWF f is called preimage-verifiable if there exists a polynomial-time computable predicate $D_f : \{0,1\}^* \longrightarrow \{0,1\}$ such that for any string y, $D_f(y) = 1$ if and only if there exists an x such that $y = f(x)$. Preimage-verifiable OWF is a generic and actually quite weak hardness assumption that includes, in particular, any certified OWP and any 1-1 length-preserving OWF. Detailed discussions on the relationship among normal OWF, preimage-verifiable OWF and 1-1 OWF are presented in Appendix D.

(2) Perfectly-binding commitment schemes: We employ both the OWP based one-round perfectly-binding commitment scheme [7, 34], and Naor's OWF-based 2-round scheme [46]. Note that the first-round message of Naor's commitment scheme can be fixed once and for all and, in particular, can be posted as part of a public-key in the public-key setting. *We remark that if the underlying OWP or OWF are secure against 2^{n^c}-time adversaries for some constant c, $0 < c < 1$, on a security parameter n, then the hiding property of the corresponding perfectly-binding commitment schemes above also holds against 2^{n^c}-time adversaries.*

(3) Public-coin witness indistinguishability (WI) proof of knowledge (POK) systems for $\mathcal{NP}$: One is Blum's protocol for directed Hamiltonian cycle DHC [8], and another is the Lapidot-Shamir protocol for DHC [41]. The salient feature of the Lapidot-Shamir protocol is that the prover sends the first-round message without knowing the statement to be proved other than its size. We remark that the WI property of Blum's protocol or the Lapidot-Shamir protocol for HC relies on the hiding property of the underlying perfectly-binding commitment scheme (used in its first-round). If the hiding property of the underlying perfectly-binding commitment scheme is secure against 2^{n^c}-time adversaries for some constant c, $0 < c < 1$, on a security parameter n, then the WI property also holds against 2^{n^c}-time adversaries.

(4) Trapdoor commitment schemes: Normal trapdoor commitment schemes run in two rounds, in which the (honest) commitment receiver generates and sends the trapdoor commitment public key ($TCPK$) in the first-round (while keeping the trapdoor secret key ($TCSK$) private). For the Feige-Shamir trapdoor commitment scheme (FSTC), $TCPK$ consists of $(y = f(x), G)$ (for OWF-based solution, the $TCPK$ also includes a random string R serving as the first-round message of Naor's OWF-based perfectly-binding commitment scheme), where f is a OWF and G is a graph that is reduced from y by the Cook-Levin $\mathcal{NP}$-reduction. The corresponding trapdoor is x (or equivalently, a Hamiltonian cycle in G). Note that the first-round message, i.e., $TCPK$, can be fixed once and for all. The commitment sender forms the second-round message by using (either OWP-based one-round or Naor's OWF-based two-round) perfectly-binding commitment scheme. *Again, if the hiding property of the underlying perfectly-binding commitment scheme is secure against sub-exponential-time adversaries, then both the hiding property and the trapdoorness property of the FSTC scheme hold also against sub-exponential-time adversaries.*

3 Concurrent Knowledge-Extractability in Public-Key Models: Motivations and Discussions

In this section we show that concurrent verifier security in public-key models is more complicated and subtle than currently known. In particular, we show that concurrent soundness is *not* sufficient for concurrent verifier security in public-key models. The underlying reason is that indeed a malicious prover cannot convince the honest verifier (with registered public-key) of a *false* statement in its concurrent interleaving interactions (just as guaranteed by concurrent soundness), **nevertheless** it may be able to convince the honest verifier of a *true* statement (in particular, verifier's public-key related one) *without knowing any corresponding $\mathcal{NP}$-witness* via concurrent interleaving interactions. For example, all known black-box rZK protocols in the BPK model run a sub-protocol in which the verifier proves to the prover the knowledge of the secret-key corresponding to its public-key. Thus, a malicious prover could potentially *malleate* the sub-protocol interactions of one session into successful interactions of

another session on a true (in particular, verifier's public-key related) statement but without knowing any corresponding $\mathcal{NP}$-witness. This potential vulnerability turns out to be a real security threat for cryptographic protocols running concurrently in public-key models (when verifiers register public-keys), in particular for rZK protocols in the BPK model when they are used as (smart-card based) *identification schemes* running over the Internet (which is just the original motivation of rZK). In particular, we present concrete attacks to natural existing ZK protocols in the BPK model that achieved concurrent soundness and are also normal arguments of knowledge. Note also that the traditional notion of proof/argument of knowledge is also originally motivated and introduced to formalize the verifier security of ZK protocols when they are used as identification schemes (as advocated in [28, 27]), showing that the traditional soundness notion is not sufficient in such cases [5].

This motivates us to introduce a stronger concurrent verifier security notion in public-key models, called (same public-key sub-exponential-time black-box) concurrent knowledge-extractability, which roughly means that a malicious prover can convince the honest verifier with its registered public-key of a statement in the concurrent interactions only if it knows a corresponding $\mathcal{NP}$-witness.

Definition 3.1 (distribution of sub-exponentially hard instances) *Let $L \in \mathcal{NP}$, and let R_L be a witness relation for L. Let $X_L \stackrel{def}{=} \{X_n\}_{n\in N}$ be a probability ensemble such that X_n ranges over $L \cap \{0,1\}^n$. We say X_L is* sub-exponentially *hard for R_L if for some constant c_L, $0 < c_L < 1$, for every sufficiently large n, and every circuit C of size at most $2^{n^{c_L}}$, $\Pr[C(X_n, 1^n) \in R_L(X_n)] < 2^{-n^{c_L}}$. We set c_L be 1 if X_L is* not *sub-exponentially hard.*

For example, if f is a (sub-exponentially strong) one-way function, then the probability ensemble $\{f(U_n)\}_{n\in N}$ is (sub-exponentially) hard for the witness relation $\{(f(x), x) : x \in \{0,1\}^*\}$, where U_n is uniform over $\{0,1\}^n$.

Definition 3.2 ((same public-key sub-exponential-time) concurrent knowledge-extractability) *We say that a protocol $\langle P, V\rangle$ for a language L is (black-box)* concurrently knowledge-extractable *in the BPK model, if there exists a sub-exponential-time (black-box) knowledge-extractor E such that for any honest verifier V with its registered public-key PK, for any sufficiently large n and any (whether false or true) sufficiently-long x of length poly(n), for all positive polynomials s and all s-concurrent malicious prover P^* (as defined in Appendix C), if P^* can convince the honest verifier (with its registered public-key PK) of the statement "$x \in L$" with non-negligible probabilities in the concurrent interactions, then the knowledge-extractor, on the* same *verifier's public-key PK with oracle accessing to P^*, will output a witness for $x \in L$ also with non-negligible probabilities* in time 2^{n^c} *for some constant c, $0 < c < c_L$, where c_L is the constant defined in Definition 3.1.*

Below, we highlight several subtle points about this definition.

The first point is that in the above definition the (black-box) knowledge extractor works in sub-exponential-time (rather than polynomial-time as in the traditional definition of argument of knowledge). The reason is that for rZK protocols we cannot count on the (*black-box*) knowledge-extractor working in polynomial-time, as rZK (black-box) arguments of knowledge exist only for $\mathcal{BPP}$ languages [3]. Therefore, we relax the definition in resettable settings by requiring the black-box knowledge-extractor to work *in sub-exponential-time*. This relaxation is justified by the fact that all known *black-box* rZK protocols in the BPK model are under sub-exponential hardness assumptions. In particular, we can assume that x is a $2^{n^{c_L}}$-hard instance for L (as defined in Definition 3.1). Now, suppose a polynomial-time concurrent malicious prover can convince the honest verifier of "$x \in L$" with non-negligible probabilities (by concurrent interleaving interactions), then the existence of the 2^{n^c}-time, $0 < c < c_L$, knowledge-extractor contradicts the underlying sub-exponential hardness assumption (or just the fact that $x \notin L$ in case $x \notin L$), which implies that a polynomial-time concurrent malicious prover can convince the honest verifier (with its public-key) of "$x \in L$" in the concurrent interactions only if it "knows" a corresponding $\mathcal{NP}$-witness. This shows, in particular, that concurrent knowledge-extractability implies (i.e., is stronger than) concurrent soundness for concurrent verifier security in public-key models.

The second point is that in our definition of concurrent knowledge-extractability, we require the knowledge-extractor to use the *same* verifier's public-key in its knowledge-extraction (i.e., without knowing the corresponding secret-key). We remark that, as we shall see in the security analyses of Section 4, same public-key knowledge-extraction with complexity leveraging (allowed by the sub-exponential assumptions) appears to be the only way at present to get around the obstacle of concurrent general composition for achieving concurrently knowledge-extractable ZK protocols in the BPK model. The underlying reason is that if the knowledge-extractor is allowed to generate simulated public-key (i.e., it would know the corresponding secret-key of the simulated public-key), and we aim for ZK protocols with black-box polynomial-time concurrent knowledge-extraction in the BPK model, then *for provable security* we might in general need to require the sub-protocol (from verifier to prover) to remain secure under concurrent general composition. The reason is that the sub-protocol (from verifier to prover) is concurrently composed with a *different* sub-protocol (from prover to verifier), and also the prover (who plays the role of verifier in the sub-protocol from verifier to prover) has no registered public-keys. The notion of concurrent general composition is introduced by Lindell [43], which considers the security of a protocol when it is run concurrently with other *different* protocols (this is in contrast to the notion of concurrent *self* composition introduced in [42] that considers the case that a *single* protocol is concurrently run many times). Concurrent general composition is a very strong security notion, which implies, in particular, concurrent non-malleability. It is shown in [43] that concurrent general composition is equivalent to a natural relaxed variant of the robust security notion of universal composability [10], and that there are large classes of two-party functionalities for which *it is impossible* to obtain protocols that remain secure under concurrent general (and even much weaker, *parallel*) composition in the standard model and even in the timing model without specified delays [37].

Thirdly, in comparison with the traditional definition of arguments of knowledge, in our definition we do not explicitly introduce the threshold (i.e., the error probability κ in Definition D.6) for knowledge-extractability. Rather, we only simply require that for any x if P^* can convince the statement "$x \in L$" with non-negligible probabilities (in its concurrent interactions with V), then knowledge-extractability will also be done with non-negligible probability. This simplified treatment suffices our purpose for showing that concurrent knowledge-extractability implies concurrent soundness.

Concurrent knowledge-extractability is strictly stronger than concurrent soundness in public-key models. To show that it is strictly stronger, we show that both the protocol of [53] (that is sequentially-sound cZK argument of knowledge in the BPK model) and the protocol of [19] (that is concurrently-sound cZK argument of knowledge) are *not* concurrently knowledge-extractable in the BPK model under any sub-exponentially strong OWP, by identifying concrete concurrent interleaving and malleating attacks that are possibly of independent interest. Actually, as clarified, *no* existing rZK or cZK protocols in the BPK model could be provably secure under this stronger concurrent verifier security. The concurrent interleaving and malleating attacks are presented in detail in Appendix E.

4 Constant-Round rZK-CKE Arguments for $\mathcal{NP}$ in the BPK model under Minimal Hardness Assumption

The high-level overview of the protocol. We first convey basic ideas and a high-level overview of the protocol. Let f_V be any (*sub-exponentially strong*) OWF, each (honest) verifier V randomly selects an element x_V from the domain of f_V, and publishes $y_V = f_V(x_V)$ as its public-key with x_V as its secret-key. Let L be an $\mathcal{NP}$-language and $x \in L$ be the common input, the main-body of the protocol goes as follows: The honest prover P first generates and sends a hard-instance using a standard *polynomially-secure* OWF f_P. The hard-instance is then fixed once and for all. Then, P proves to V the existence of the preimage of the hard-instance, by executing a OWF-based resettable witness-hiding (rWH) protocol. After that, V proves to P that it knows either the preimage of y_V (i.e., its secret-key x_V) or the preimage of the hard-instance generated by P, by executing a OWF-based constant-round WIPOK protocol for $\mathcal{NP}$. Finally, P proves to V that it knows either a witness for $x \in L$ or the preimage of y_V (i.e., V's secret-key), by executing another OWF-based constant-round rWI argument

for $\mathcal{NP}$. The detailed protocol description is depicted in Figure 1 (page 7).

The underlying complexity-leveraging. For this protocol to be provably secure, we employ the complexity-leveraging technique (that is originally introduced in [11] and used in all previous *black-box* rZK systems in the BPK model). Specifically, the verifier V uses a security parameter N (in generating messages from it) that is also the system security parameter. But, the prover P uses a relatively smaller security parameter n (that is still polynomially related to N). The justification and discussions of the complexity-leveraging technique are given in [11]. *Here, we additionally remark that, pragmatically speaking, letting the verifier and the prover use different security parameters is quite reasonable in the resettable setting, in which the prover is implemented by smart-cards or clients that have relatively limited computational resources and power and the verifier is normally implemented by servers that have much more computational resources and power.*

Specifically, the security parameters are set as follows. On the system parameter N, suppose f_V is secure against $2^{N^{c_V}}$-time adversaries for some constant c_V, $0 < c_V < 1$. And for any $x \in L \cap \{0,1\}^{poly(N)}$, let c_L, $0 < c_L \leqslant 1$, be the constant defined in Definition 3.1. Let c be any constant such that $0 < c < \min\{c_V, c_L\}$ (in other words, $\min\{c_V, c_L\} = c + c'$ for another constant c', $0 < c' < 1$). The prover uses a relatively smaller security parameter n and uses a standard polynomially-secure OWF f_P that can be broken (brute-force wise) in time $2^{n^{c_P}}$ for some constant c_P, $c_P \geqslant 1$. Let ε be any constant such that $\varepsilon > \frac{c_P}{c}$, then we set $N = n^{\varepsilon}$. Note that N and n are polynomially related. That is, any quantity that is a polynomial of N is also (another) polynomial of n. This complexity leveraging guarantees that although any $poly(n) \cdot 2^{n^{c_P}}$-time adversary can break f_P on a security parameter n, it is still infeasible to break the one-wayness of f_V (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$). Also note that any $poly(n) \cdot 2^{n^{c_P}}$-time algorithm cannot output a witness for $x \in L$ with non-negligible probabilities, in case x is a sub-exponentially hard instance or just $x \notin L$ (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_L}}$). However, we show that for any (whether true or not) common input $x \in \{0,1\}^{poly(N)}$, if a PPT concurrent malicious P^* can convince V of the statement "$x \in L$" with non-negligible probabilities in its concurrent interactions, then there exists a black-box knowledge-extractor that, on input y_V (i.e., V's public-key), works in $poly(n) \cdot 2^{n^{c_P}}$-time and outputs a witness for $x \in L$ also with non-negligible probabilities. Thus, under reasonable (i.e., sub-exponential) hardness assumptions on the language L, no PPT concurrent malicious prover can convince V of any (sufficiently long) statement without "knowing" a witness.

The OWF-based protocol depicted in Figure 1 (page 7) runs in 7 rounds after some round combinations. In particular, the first two rounds of Phase-4 can be combined into previous phases. Actually, as we shall see in Section 5, the round-complexity can be further reduced to six but under any (sub-exponentially strong) OWP. Now, for the protocol depicted in Figure-1, we have the following theorem:

Theorem 4.1 *Assuming the OWF f_P (used by the prover) is secure against standard polynomial-time adversaries, and the OWF f_V (used by the verifier) is secure against sub-exponential-time adversaries, the protocol depicted in Figure-1 is a constant-round concurrently knowledge-extractably secure rZK (rZK-CKE) argument for $\mathcal{NP}$ in the BPK model.*

Proof (sketch). We present a brief sketch of the proof of Theorem 4.1; a detailed proof is given in Appendix F.

Black-box resettable zero-knowledge.

For any s-resetting adversary V^* (as defined in Appendix C) who receives $s(N)$ *distinct* strings $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(N)}\}$, $x_i \in L \cap \{0,1\}^{poly(N)}$ for each i ($1 \leqslant i \leqslant s(N)$), and outputs an arbitrary public-file F containing $s(N)$ entries $PK_1, \cdots, PK_{s(N)}$ in its first stage, we say a public-key PK_j in F, $1 \leq j \leq s(N)$, is "covered" if the rZK simulator S has already learnt (extracted) the corresponding secret-key SK_j (if such exists). In its second stage, V^* is given oracle access to $(s(N))^3$ prover instances $P(x_i, PK_j, \gamma_k)$, $1 \leqslant i, j, k \leqslant s(N)$. We denote by $D_t = (x_i, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, (c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t)$ the "determining" message of the t-th session with respect to common input x_i and public-key PK_j and the honest prover instance $P(\cdot, \cdot, \gamma_k)$, $1 \leq i, j, k \leq s(N)$ and $1 \leqslant t \leqslant (s(N))^3$. As discussed in Appendix C

Key generation. Let f_V be any OWF that is secure against $2^{N^{c_V}}$-time adversaries. On a security parameter N, each honest verifier V randomly selects an element x_V of length N, computes $y_V = f_V(x_V)$, publishes y_V as its public-key PK while keeping x_V as its secret-key SK. (If P uses Naor's OWF-based perfectly-binding commitment scheme in Phase-2 or Phase-4 (that is run on security parameter n), V also deposits a random string R_V of length $3n$ as a part of its public-key serving as the first-round message of Naor's commitment scheme used by P.)

Common input. An element $x \in L \cap \{0,1\}^{poly(N)}$ (let c_L, $0 < c_L \leqslant 1$, be the constant defined in Definition 3.1.), the public-file F and an index j that specifies the j-th entry of F, i.e., $PK_j = (y_V^{(j)}, R_V^{(j)})$. Note that the system security parameter is N.

P **private input.** An $\mathcal{NP}$-witness w for $x \in L$, a pair of random strings (γ_1, γ_2), where γ_1 is a $poly(n)$-bit string and γ_2 is an n-bit string serving as the randomness seed of a PRF.

V **private input.** SK_j. For simplicity of presentation, except explicitly clarified we denote $PK_j = f_V(SK_j)$.

Complexity-leverage used. Let c_V, $0 < c_V < 1$, be the constant that the one-wayness of the OWF f_V, and thus the hiding property of the underlying perfectly-binding commitment scheme used by the verifier all hold against any circuit of size $2^{N^{c_V}}$ (which in turn guarantees that the WI property of the underlying WI protocol for $\mathcal{NP}$ executed in Stage-2 of Phase-1 and Phase-3, the hiding and trapdoorness properties of the underlying trapdoor commitment scheme all hold against any circuit of size $2^{N^{c_V}}$). The prover uses a relatively smaller security parameter n. Let c_P, $c_P \geqslant 1$, be the constant that: for all sufficiently large n's, the size of G_P (reduced from $(y_P^{(0)}, y_P^{(1)})$) is bounded by n^{c_P}, which in turn implies that the perfectly-binding commitment scheme used by the prover (that is run on the security parameter n) can be brute-force decommitted in time $2^{n^{c_P}}$. Let c be any constant such that $0 < c < \min\{c_V, c_L\}$ and ε be any constant such that $\varepsilon > \frac{c_P}{c}$, then we set $N = n^{\varepsilon}$.

Phase-1. Phase-1 consists of two stages:

Stage-1. Let f_P be any (polynomially-secure) OWF. On security parameter n, P randomly selects two elements $x_P^{(0)}$ and $x_P^{(1)}$ of length n each in the domain of f_P, computes $y_P^{(b)} = f_P(x_P^{(b)})$ for $b \in \{0,1\}$, reduces $(y_P^{(0)}, y_P^{(1)})$ to a directed graph G_P by Cook-Levin $\mathcal{NP}$-reduction such that finding a Hamiltonian cycle in G_P is equivalent to finding the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$. For OWF-based solution, P also randomly selects a string R_P of length $3N$ serving as the first-round message of Naor's OWF-based perfectly-binding commitment scheme. Finally, P sends $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)$ to V. The randomness used by P in this process is γ_1, *which means that the* $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)$ *is fixed once and for all.*

Stage-2. V first checks whether or not G_P is reduced from $(y_P^{(0)}, y_P^{(1)})$ and R_P is of length $3N$. If the checking is successful, V randomly chooses two random strings $e_V^{(0)}$ and $e_V^{(1)}$ from $\{0,1\}^n$, computes $c_V^{(0)} = Com(1^N, R_P, e_V^{(0)})$ *by using the underlying perfectly-binding commitment scheme* Com, and $c_V^{(1)} = TCCom(1^N, (G_P, R_P), e_V^{(1)})$ by using the underlying trapdoor commitment scheme. Then, on common input $((y_P^{(0)}, y_P^{(1)}, G_P, R_P), PK_j)$ V computes the first-round message, denoted a_V, of (n-parallel repetitions of) Blum's WIPOK for $\mathcal{NP}$ for showing the knowledge of either SK_j or a Hamiltonian cycle in G_P (equivalently, the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$). Finally, V sends $(c_V^{(0)}, c_V^{(1)}, a_V)$ to P. *From then on, all randomness used by* P *in the remaining computation is got by applying* $PRF(\gamma_2, \cdot)$ *on the "determining" message* $D = (x, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P), (c_V^{(0)}, c_V^{(1)}, a_V))$.

Phase-2. P proves to V the existence of a Hamiltonian cycle in G_P (equivalently, the existence of the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$) by executing the (n-parallel repetitions of) Blum's WI protocol for $\mathcal{NP}$ on common input $(y_P^{(0)}, y_P^{(1)}, G_P, R_V^{(j)})$, in which V sends the assumed random challenge by just revealing $e_V^{(0)}$ committed to $c_V^{(0)}$. Note that the first-round message of Phase-2 (from P to V) consists of n committed adjacency matrices committed by running the underlying perfectly-binding commitment scheme *on security parameter* n. If P successfully finishes this phase and V accepts, then goto Phase-3. Otherwise, V aborts.

Phase-3. V and P continue the WIPOK protocol for $\mathcal{NP}$ suspended at Stage-2 of Phase-1. If V successfully convinces P of the knowledge of either SK_j or a Hamiltonian cycle in G_P, then goto Phase-4. Otherwise, P aborts. We denote by e_V, z_V, the first-round message and the second-round message of Phase-3 respectively.

Phase-4. P proves to V that it "knows" either the witness w for $x \in L$ or the secret-key SK_j, by executing the (n-parallel repetitions of) Blum's WI protocol for $\mathcal{NP}$ on common input (x, PK_j), in which V sends the assumed random challenge by just revealing $e_V^{(1)}$ committed to $c_V^{(1)}$.

Figure 1. Constant-round rZK-CKE arguments for $\mathcal{NP}$ under the minimal hardness assumption

and F, wlog, we assume V^* works in the sequential version in its second stage and the honest prover P utilizes a truly random function rather than a pseudorandom function.

The rZK simulation procedure is similar to, but more complicated than that of [11]. Specifically, the rZK simulator S runs V^* as a subroutine, and works in at most $s(N)+1$ phases such that in each phase

it either successfully finishes its simulation or "covers" a new public-key in F. In each phase, S makes a simulation attempt from scratch with a new truly random function that is to be defined adaptively, and works session by session sequentially in at most $(s(N))^3$ sessions. The difficulties lie in that for such rZK simulation to be successful, the rZK simulator S needs to have the ability to cover new uncovered public-keys within time inversely proportional to the probability that it encounters a success of Phase-3 relative to a yet uncovered public-key in its simulation, as pending on S's such ability the rZK property is from the rWI property of Phase-4 combined with Phase-1.

Specifically, we want to argue that the underlying Blum's WIPOK protocol on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ (executed in Stage-2 of Phase-1 together with Phase-3) is actually an *argument of knowledge* of the preimage of PK_j (i.e., the secret-key SK_j). But, the subtle and complicated situation here is that before V^* finishes Phase-3, S has already proved the knowledge of the Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ in Phase-2. Note that the $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ is fixed once and for all (which can be viewed as the public-key of the honest prover instance $P(\cdot, \cdot, \gamma_k)$), and furthermore V^* is resettingly (more than concurrently) interacting with the honest prover instances. As demonstrated in Section 3 and Appendix E, normal argument of knowledge and even concurrent soundness do not guarantee knowledge-extractability in such a setting. In particular, one may argue that, by rewinding the honest prover instances arbitrarily, V^* may potentially forge the interactions on $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ provided by the honest prover in Phase-2 of one session into successful but "false" interactions on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ in Stage-2 of Phase-1 and Phase-3 of another session with respect to public-key PK_j, in the sense that although the interactions are valid but V^* actually does not know the corresponding secret-key SK_j. This means that, in such a case the interactions on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ executed in Phase-3 together with Stage-2 of Phase-1 are not any longer an *argument of knowledge* of the preimage of PK_j, although it is always a system for proof of knowledge of either SK_j or a Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$. What save us here is the (concurrent) WI property of the Blum's protocol for HC.

Below, we construct an algorithm $\hat{S}$ that emulates the real rZK simulator while *concurrently* (not resettingly) running the Blum's protocol for HC. That is, on common inputs $\{(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^1, \cdots, (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^{s(N)}\}$ $\hat{S}$ concurrently interacts with $s(N)$ instances of the knowledge prover, denoted $\hat{P}$, of Blum's protocol for HC by playing the role of knowledge verifier. We denote each of the $s(N)$ instances of $\hat{P}$ by $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$, $1 \leqslant k \leqslant s(N)$; At the same time, $\hat{S}$ runs the s-resetting malicious V^* as a subroutine by playing the role of the honest prover, and sends $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ as the Stage-1 message of Phase-1 whenever V^* initiates a session with the honest prover instance $P(\cdot, \cdot, \gamma_k)$. $\hat{S}$ emulates the rZK simulator S but with the following modification: whenever $\hat{S}$ needs to send a "fresh" first-round message of Blum's protocol for HC on $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ in Phase-2 with respect to a "determining" message, it initiates a new session with $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$, and forwards the first-round message received from $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$ to V^*. This "fresh" message happens due to either V^* sends a distinct "determining" message in one session or $\hat{S}$ needs to rewind V^* and to redefine the underlying random function f to extract knowledge used by V^* in a successful execution of Stage-2 of Phase-1 and Phase-3 with respect to an uncovered public-key. Then, $\hat{S}$ runs V^* further, and in case V^* successfully reveals the assumed challenge (that is *perfectly-bindingly* committed to the underlying "determining" message in question) then $\hat{S}$ returns back the revealed challenge to $\hat{P}$ as its own challenge in the corresponding simultaneous session of Blum's protocol for HC, and returns back the third-round message received from $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$ to V^*. For a session with a "determining" message that is identical to that of some previous sessions, $\hat{S}$ just copies what was sent in the previous sessions. Note that in this case, $\hat{S}$ may still possibly need to interact with $\hat{P}$ in some *existing* concurrent session to get some third-round message (in case V^* did not reveal or invalidly revealed the random challenge perfectly-bindingly committed to the underlying "determining" message in all previous sessions, but correctly reveals it in the current session). However, the key point here is that in this case S does not need to initiate a new concurrent session with $\hat{P}$.

Note that from the viewpoint of V^*, the behavior of $\hat{S}$ is identical to the behavior of the real rZK

simulator, where the real rZK simulator S generates $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$'s and provides the corresponding Phase-2 messages by itself (rather than get it by interacting with the knowledge prover instances $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$'s of the Blum's protocol for HC). The key observation here is that although V^* is actually resettingly interacting with $\hat{S}$, $\hat{S}$ only concurrently interacts with the instances of $\hat{P}$ and never rewinds $\hat{P}$. The underlying reason is just that in any session, Phase-2 interactions take place only after V^* sent the "determining" message at Stage-2 of Phase-1 that determines the subsequent behaviors of V^* in that session. Note that in this case, the (concurrent) WI property of the Blum's protocol for HC on common input $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ actually implies witness hiding (WH), which means that no PPT algorithm can output a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ even by concurrently interacting with $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$. Also note that on common input $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$, Phase-3 together with Stage-2 of Phase-1 is always a system for proof of knowledge of either a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ or the preimage of PK_j (i.e., SK_j), which means that with overwhelming probabilities $\hat{S}$ (or the real rZK simulator S) can always extract either a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ or the corresponding secret-key SK_j within time inversely proportional to the probability that V^* successfully finishes Phase-3 (by rewinding V^* and redefining the underlying random function). But, the WH property of Blum's protocol for HC shows that with overwhelming probabilities, $\hat{S}$ (or the real rZK simulator S) never outputs a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ in its simulation that is done in expected polynomial-time. Here, a subtle point needs to be further addressed. Specifically, the normal WH property is defined with respect to probabilistic (strict) polynomial-time algorithms, but here $\hat{S}$ works in expected polynomial-time. But, by Markov inequality, it is easy to see that if the WH property of a protocol holds with respect to any strict polynomial-time algorithms, then it also holds with respect to any expected polynomial-time algorithms. Details are given in Appendix F.

Comments: Note that in the proof of rZK, we require nothing about the public-keys registered by V^* in F. What we need in the simulation is the POK property of Blum's protocol executed in Stage-2 of Phase-1 and Phase-3, which does hold with respect to *any* common input (in particular, *any* public-key registered by V^*, whether valid or not). That is, our protocol is not limited to the structure of the keys. Also note that for the OWF f_P used by the prover, it suffices that it is secure against standard polynomial-time adversaries, since the one-wayness of f_P is only used to guarantee the computationally-binding property of the underlying FSTC scheme against malicious polynomial-time verifiers (in proving black-box resettable zero-knowledge). Similarly, the pseudorandoness of the PRF and the hiding property of the perfectly-binding commitment scheme used by the prover (that are based on f_P) can be similarly secure against polynomial-time adversaries.

Black-box concurrent knowledge-extractability.

We show that for any (whether true or not) common input $x \in \{0,1\}^{poly(N)}$, if a PPT s-concurrent malicious P^* can convince an honest verifier V (with public-key PK and secret-key SK) of the statement "$x \in L$" with non-negligible probability p_x in one of the $s(N)$ concurrent interactions, then there exists a black-box knowledge-extractor E that, on input PK with oracle accessing P^*, works in $poly(n) \cdot 2^{n^{c_P}}$-time and outputs a witness for $x \in L$ also with non-negligible probabilities. Note that according to the underlying complexity leveraging on the security parameters N and n, no $poly(n) \cdot 2^{n^{c_P}}$-time algorithm can break the one-wayness of f_V used by V in forming its public-key on security parameter N (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$). Also note that any $poly(n) \cdot 2^{n^{c_P}}$-time algorithm cannot output a witness for $x \in L$ with non-negligible probabilities, in case x is a sub-exponentially hard instance or just $x \notin L$ (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_L}}$).

Taking PK as its input, E randomly chooses j from $\{1, \cdots, s(N)\}$ and runs P^* as a subroutine by playing the role of the honest verifier with public-key PK. *Note that E does not know the corresponding secret-key SK.* In each session t, $1 \leqslant t \leqslant s(N)$, after receiving the Stage-1 message of Phase-1, denoted $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^t, G_{P^*}^t, R_{P^*}^t)$, E first checks whether or not $G_{P^*}^t$ is $\mathcal{NP}$-reduced from $(y_{P^*}^{(0)}, y_{P^*}^{(1)})^t$ and $R_{P^*}^t$ is of length $3N$. If the checking is successful, then E tries to find a Hamiltonian cycle in $G_{P^*}^t$ by brute-force searching in $2^{n^{c_P}}$-time.

- If E finds a Hamiltonian cycle in $G_{P^*}^t$, then E sets the Stage-2 message of Phase-1 of the t-th

session, denoted $((c_V^{(0)})^t, (c_V^{(1)})^t, a_V^t)$, as follows: it randomly chooses one random string $(e_V^{(0)})^t$ from $\{0,1\}^n$, computes $(c_V^{(0)})^t = Com(1^N, R_{P^*}^t, (e_V^{(0)})^t)$ by using the underlying Naor's perfectly-binding commitment scheme Com, and computes $(c_V^{(1)})^t = TCCom(1^N, (G_{P^*}^t, R_{P^*}^t), 0^n)$ by using the underlying Feige-Shamir trapdoor commitment scheme (note that, $(c_V^{(1)})^t$ commits to 0^n rather than a random string in $\{0,1\}^n$ as the honest verifier does). Then, on common input $(((y_P^{(0)}, y_P^{(1)})^t, G_{P^*}^t, R_{P^*}^t), PK)$ V computes the first-round message, denoted a_V^t, of (n-parallel repetitions of) Blum's WIPOK for $\mathcal{NP}$ for showing the knowledge of either SK or a Hamiltonian cycle in $G_{P^*}^t$. *Note that the first-round message of Blum's WIPOK for $\mathcal{NP}$ is computed without using any witness knowledge (i.e., either SK or a Hamiltonian cycle in $G_{P^*}^t$)*; In case P^* successfully finishes Phase-2 of the t-th session, E moves into Phase-3. After receiving the first-round message of Phase-3 of the t-th session, denoted e_V^t, E computes the second-round message of Phase-3, denoted z_V^t (i.e., the third-round message of Blum's WIPOK for showing the knowledge of either SK or a Hamiltonian cycle in $G_{P^*}^t$), *by using the extracted Hamiltonian cycle in $G_{P^*}^t$ as its witness*; Finally, in Phase-4 of the t-th session, E decommits $(c_V^{(1)})^t$ to a random string $(e_V^{(1)})^t$ of length n, *by using the extracted Hamiltonian cycle in $G_{P^*}^t$ as the trapdoor.*

- If there exists *no* Hamiltonian cycle in $G_{P^*}^t$, then E sets and sends the Stage-2 message of Phase-1 of the t-th session, i.e., $((c_V^{(0)})^t, (c_V^{(1)})^t, a_V^t)$, just as above. But, whenever P^* successfully finishes Phase-2 of the t-th session and sends to E the first-round message of Phase-3 of the t-th session (i.e., e_V^t), E aborts with an error message (as it has no witness for generating the next message).

In the j-th session with respect to a common input x_j selected by P^* on the fly, we denote by $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^j, G_{P^*}^j, R_{P^*}^j)$ the Stage-1 message of Phase-1 of this session, and by $((c_V^{(0)})^j, (c_V^{(1)})^j, a_V^j)$ the Stage-2 message of Phase-1 of this session that is set as specified above (where $(c_V^{(1)})^j = TCCom(1^N, (G_{P^*}^j, R_{P^*}^j), 0^n)$). In case P successfully finishes this session (i.e., Phase-4 of the j-th session), we denote by $a_{P^*}^j$ the first-round message of Phase-4 of the j-th session, by $(e_V^{(1)})^j$ the random challenge (of length n) sent by E in the second-round of Phase-4 (by decommitting $(c_V^{(1)})^j$ with the extracted Hamiltonian cycle in $G_{P^*}^j$ as the trapdoor), and by $z_{P^*}^j$ the third-round message of Phase-4 of the j-th session, where $(a_{P^*}^j, (e_V^{(1)})^j, z_{P^*}^j)$ constitutes a successful conversation of (n-parallel repetitions of) Blum's WIPOK for showing the knowledge either a witness for $x_j \in L$ or the corresponding secret-key SK. Then, after receiving the last-round message (i.e., $z_{P^*}^j$), E rewinds P^* to the state that it just sent $a_{P^*}^j$, decommits $(c_V^{(1)})^j$ to a different random string $(e_V^{(1)})^{j'}$ (that is taken uniformly from $\{0,1\}^n / \{(e_V^{(1)})^j\}$)) by using the extracted Hamiltonian cycle in $G_{P^*}^j$ as the trapdoor, and runs P^* further, expecting to receive another valid third-round message, denoted $z_{P^*}^{j'}$, of Phase-4 of the j-th session.

For any x, denote by q_x the probability that P^* successfully convinces $E(PK)$ of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions (without rewinding in the j-th session). Then, with probability about $\frac{(q_x)^2}{s(N)}$ E will output either a witness w for $x \in L$ or the corresponding secret-key SK such that $PK = f_V(SK)$ in $poly(n) \cdot 2^{n^{c_P}}$-time, by rewinding the j-th session for a randomly chosen j from $\{1, \cdots, s(N)\}$. As $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$ and f_V is secure against any circuit of size $2^{N^{c_V}}$, we know with probability negligibly close to $\frac{(q_x)^2}{s(N)}$ E will output a witness w for $x \in L$. Now, to establish the concurrent knowledge-extractability property, all that is left is to show that $|p_x - q_x|$ is negligible *for any* x, where p_x is the probability that P^* successfully convinces the honest verifier with public-key PK of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions. This is done by establishing a series of hybrid experiments.

We first consider a mental experiment in which P^* concurrently interacts with an imaginary verifier $\widehat{V}$ with the same public-key PK and secret-key SK. $\widehat{V}$ mimics the real honest verifier V with public-key PK and secret-key SK but with the following modifications: For any session t, $1 \leqslant t \leqslant s(N)$, in case P^* successfully finishes Phase-2 and sends to $\widehat{V}$ the first-round message of Phase-3, $\widehat{V}$ enumerates all possible Hamiltonian cycles of $G_{P^*}^t$ by brute-force searching in $2^{n^{c_P}}$-time, where $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^t, G_{P^*}^t, R_{P^*}^t)$

is the Stage-1 message of Phase-1 of the t-th session. If there exists *no* Hamiltonian cycle in $G^t_{P^*}$, then $\widehat{V}$ aborts with an error message (*although it can continue the execution with SK as its witness*).

For any x, denote by $\hat{p}_x$ the probability that P^* successfully convinces the imaginary verifier $\widehat{V}$ with public-key PK of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions. We want to show that for any x $|p_x - \hat{p}_x|$ is negligible in n. Note that the only difference between the interactions between P^* and $\widehat{V}$ and the interactions between P^* and the real honest verifier V is that: for any session t, $1 \leqslant t \leqslant s(N)$, the real honest verifier always continues the execution of Phase-3 by using SK as its witness in forming the second-round message of Phase-3, in case P^* successfully finished Phase-2 and sent the first-round message of Phase-3 ; but $\widehat{V}$ may abort in this case if it finds that $G^t_{P^*}$ is "false" (i.e. there exists no Hamiltonian cycle in $G^t_{P^*}$) by brute-force searching in $2^{n^{c_P}}$-time. Thus, the fact that for any x $|p_x - \hat{p}_x|$ is negligible is derived from the following lemma.

Lemma 4.1 *For all positive polynomials $s(\cdot)$ and all s-concurrent malicious P^*, the probability that there exists a t, $1 \leqslant t \leqslant s(N)$, such that P^* can successfully finish Phase-2 with respect to a false $G^t_{P^*}$ (i.e., $G^t_{P^*}$ contains no Hamiltonian cycle) in the t-th session of the $s(N)$ concurrent sessions (against the real honest verifier V with public-key PK) is negligible in n. Note that any quantity that is negligible in n is also negligible in N.*

Proof (of Lemma 4.1). We show that if a PPT s-concurrent adversary P^* can convince V (with public-key PK) of a false $G^t_{P^*}$ with non-negligible probability $p'(n)$ in Phase-2 of one of the $s(N)$ concurrent sessions, then this will violate the hiding property of the underlying *perfectly-binding* commitment scheme, denoted Com, used by V in Phase-1 that is run on security parameter N. Note that according to the hiding property of the underlying perfectly-binding commitment scheme Com, given two strings $\hat{e}_0$ and $\hat{e}_1$ that are taken uniformly at random from $\{0,1\}^n$ and $C = Com(1^N, R^t_{P^*}, \hat{e}_b)$ for a randomly chosen bit $b \in \{0,1\}$, no $2^{N^{c_V}}$-time (non-uniform) algorithm can distinguish whether C commits to $\hat{e}_0$ or to $\hat{e}_1$ (i.e., guess the bit b correctly) with non-negligible advantage over $1/2$, even with $\hat{e}_0$, $\hat{e}_1$ and the secret-key of V (i.e., SK) as its non-uniform inputs.

We construct a (non-uniform) algorithm A that takes $(1^n, (\hat{e}_0, \hat{e}_1, SK), C)$ as input and attempts to guess b with a non-negligible advantage over $1/2$, where $\hat{e}_0$ and $\hat{e}_1$ are taken uniformly at random from $\{0,1\}^n$ and $C = Com(1^N, R_{P^*}, \hat{e}_b)$ for a randomly chosen bit $b \in \{0,1\}$. E randomly selects j from $\{1, \cdots, s(N)\}$, runs P^* as a subroutine by playing the role of the honest verifier V with secret-key SK in any session other than the j-th session. In the j-th session, after receiving $G^j_{P^*}$ from P^* at Stage-1 of Phase-1, E first checks whether there exists a Hamiltonian cycle in $G^j_{P^*}$ or not by brute-force searching in time $2^{n^{c_P}}$. If E finds a Hamiltonian cycle in $G^j_{P^*}$, then E randomly guesses the bit b and stops. Otherwise (i.e., there exists no Hamiltonian cycle in $G^j_{P^*}$), E runs P^* further and continues the interactions of the j-th session as follows: E gives C to P^* as the assumed commitment to $(e^{(0)}_V)^j$ at Stage-2 of Phase-1. After receiving the first-round message of Phase-2 (i.e., the first-round of Blum's protocol for proving the existence of a Hamiltonian cycle in $G^j_{P^*}$) that contains n committed adjacency matrices, E first opens all the committed adjacency matrices by brute-force in $poly(n) \cdot 2^{n^{c_P}}$-time (note that E can do this since the underlying perfectly-binding commitment scheme used by the prover in forming these n committed adjacency matrices is run on security parameter n). For each revealed graph G^j_k $(1 \leq k \leq n)$ (described by the corresponding opened adjacency matrix entries) we say that G^j_k is a 0-valid graph if it is isomorphic to $G^j_{P^*}$, or a 1-valid graph if it contains a Hamiltonian cycle of the same size of $G^j_{P^*}$. We say that the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_b$-valid ($b \in \{0,1\}$) if for all k, $1 \leq k \leq n$, G^j_k is a $\hat{e}^{(k)}_b$-valid graph, where $\hat{e}^{(k)}_b$ denotes the k-th bit of $\hat{e}_b$. Note that for the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$, E can determine whether it is $\hat{e}_0$-valid or $\hat{e}_1$-valid in time $poly(n) \cdot 2^{n^{c_P}}$. Then, E outputs 0 if the set $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_0$-valid but not $\hat{e}_1$-valid. Similarly, E outputs 1 if the set $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_1$-valid but not $\hat{e}_0$-valid. In other cases, E just randomly guesses the bit b.

The key observation here is that if $G^j_{P^*}$ is false (i.e., containing no Hamiltonian cycle), then for each revealed graph it cannot be both a 0-valid graph and a 1-valid graph. Similarly, for false $G^j_{P^*}$, the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ cannot be both $\hat{e}_0$-valid and $\hat{e}_1$-valid for different $\hat{e}_0 \neq \hat{e}_1$. Furthermore,

suppose C commits to $\hat{e}_b$ ($b \in \{0,1\}$), then for false $G^j_{P^*}$ with probability $1-2^{-n}$ the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ is not $\hat{e}_{1-b}$-valid (since $\hat{e}_{1-b}$ is taken uniformly at random from $\{0,1\}^n$). Since the value j is randomly chosen from $\{1, \cdots, s(N)\}$, we conclude that E can successfully guess the bit b with probability at least $(1-2^{-n}) \cdot \frac{p'(n)}{s(N)} + \frac{1}{2}(1-\frac{p'(n)}{s(N)}) = \frac{1}{2} + \frac{1}{2} \cdot \frac{p'(n)}{s(N)} - 2^{-n} \cdot \frac{p'(n)}{s(N)}$ in time $poly(n) \cdot 2^{n^{c_P}}$. That is, E successfully guesses the bit b with non-negligible advantage over $1/2$ in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$, which violates the hiding property of the underlying perfectly-binding commitment scheme Com used by V that is run on the security parameter N. □

After establishing that for any x $|p_x - \hat{p}_x|$ is negligible, to show that for any x $|p_x - q_x|$ is negligible, we can show that for any x $|\hat{p}_x - q_x|$ is negligible. This is done by conducting another hybrid experiment.

Specifically, we consider another hybrid experiment, in which a PPT algorithm $\widehat{E}$ takes (PK, SK) as its input (that is, $\widehat{E}$ takes both the verifier's public-key and the corresponding secret-key as its input), and runs P^* as a subroutine by mimicking the knowledge-extractor E (which only takes PK as input) but with the following modification: For any session t, $1 \leqslant t \leqslant s(N)$, in case P^* successfully finishes Phase-2 and sends to $\widehat{E}$ the first-round message of Phase-3, $\widehat{E}$ enumerates all possible Hamiltonian cycles of $G^t_{P^*}$ by brute-force searching in $2^{n^{c_P}}$-time, where $((y^{(0)}_{P^*}, y^{(1)}_{P^*})^t, G^t_{P^*}, R^t_{P^*})$ is the Stage-1 message of Phase-1 of the t-th session. If there *exists* a Hamiltonian cycle in $G^t_{P^*}$, then $\widehat{E}$ continues the execution by forming the second-round message of Phase-3 of the t-th session (for showing the knowledge of either SK or a Hamiltonian cycle of $G^t_{P^*}$) *but using SK as its witness just as the real honest verifier does* (note that in this case E continues the execution with the extracted Hamiltonian cycle of $G^t_{P^*}$ as the corresponding witness). If there exists *no* Hamiltonian cycle in $G^t_{P^*}$, then $\widehat{E}$ aborts with an error message just as E (or $\widehat{V}$) does (*although in this case $\widehat{E}$ can continue the execution with SK as its witness*).

Note that the difference between the interactions between P^* and the imaginary verifier $\widehat{V}$ in the first hybrid experiment and the interactions between P^* and $\widehat{E}$ is that: in any session t, $1 \leqslant t \leqslant s(N)$, of the interactions between P^* and $\widehat{V}$, $\widehat{V}$ always commits (and accordingly decommits to) a random string of length n (i.e., $(e^{(1)}_V)^t$) by using the underlying FSTC scheme (just as the honest verifier V does), but in the interactions between P^* and $\widehat{E}$, $\widehat{E}$ always commits 0^n and then decommits to a random string of length n by using the brute-force extracted Hamiltonian cycle of $G^t_{P^*}$ as the trapdoor (just as E does). The difference between the interactions between P^* and $\widehat{E}$ and the interactions between P^* and E is that: E always uses the brute-force extracted Hamiltonian cycle of $G^t_{P^*}$ as its witness in Phase-3 of any session t, $1 \leqslant t \leqslant s(N)$, but $\widehat{E}$ always uses the verifier's secret-key SK as its witness (just as the honest verifier does).

For any x, denote by $\hat{q}_x$ the probability that P^* can convince $\widehat{E}$ of the statement "$x \in L$" in one of the $s(N)$ sessions. Then if there exists an x such that $|\hat{p}_x - \hat{q}_x|$ is non-negligible, we can break the hiding and trapdoorness properties of the underlying FSTC scheme in the following way: We construct a (non-uniform) algorithm $\widehat{A}$ that takes (x, PK, SK) as the (non-uniform) input and runs P^* as a subroutine by emulating either $\widehat{V}$ or $\widehat{E}$ (Note that either $\widehat{V}$ or $\widehat{E}$ can be emulated in $poly(n) \cdot 2^{n^{c_P}}$-time). Whenever $\widehat{A}$ finds that P^* successfully convinces of the statement "$x \in L$" $\widehat{A}$ outputs 1, otherwise $\widehat{A}$ outputs 0. Clearly, by standard hybrid technique, if $|\hat{p}_x - \hat{q}_x|$ is non-negligible we can break the hiding and trapdoorness properties of the underlying FSTC scheme in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$. Similarly, we can also prove that for any x $|\hat{q}_x - q_x|$ is negligible, as otherwise we can break the WI property of Blum's protocol for $\mathcal{NP}$ in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$. Thus, we get that for any x $|\hat{p}_x - q_x|$ is negligible, and so is $|p_x - q_x|$ for any x because we have shown that for any x $|p_x - \hat{p}_x|$ is negligible. This finishes the proof for concurrent knowledge-extractability of the protocol depicted in Figure-1 in the BPK model. □

Constant-round rWI arguments for $\mathcal{NP}$ under minimal hardness assumptions in the standard model. The proof of Theorem 4.1 implies that the protocol depicted in Figure-1 contains a sub-protocol, specifically the combination of Phase-1 and Phase-2, that is constant-round rWI *argument* for $\mathcal{NP}$ *in the standard model* under minimal hardness assumptions, a result unknown previously that is of independent interest. We remark that the construction of the rWI argument is actually conceptually

simple, but as we have seen, the security analyses are however complicated and subtle. Details of the following result are given in Appendix G.

Corollary 4.1 *Under any (sub-exponentially strong) OWF (resp. OWP), any language in $\mathcal{NP}$ has a 5-round (resp. 4-round) rWI argument in the standard model.*

5 Simplified and Round-Optimal Implementations

We briefly discuss simplified and round-optimal implementations of rZK-CKE arguments for $\mathcal{NP}$ in the BPK model, the details are given in Appendix H (or see [51] where we give more details but in the context of concurrent soundness, though the proofs presented in [51] actually work in the concurrent knowledge extractability setting; we plan to modify and expand [51] to be an extended version of the current submission).

Zap-based simplified implementation. A natural way to simplify the implementation of the protocol depicted in Figure-1, while maintaining almost the same protocol structure, is to replace Blum's WI protocol (with commitment of the random challenge, i.e., $e_V^{(0)}$, on the top) executed in Phase-1 and Phase-2 by *zap* developed in [23]. Zap is itself a 2-round public-coin WI *proof* for $\mathcal{NP}$. It can be easily modified into a 2-round rWI *proof* for $\mathcal{NP}$, by applying a PRF on the first-round message, denoted ξ, and the common input to get the randomness for generating the second-round message, denoted Π [23]. Furthermore, the first-round message of a zap can be fixed once and for all, and thus can be posted as a part of the underlying public-key. The zap-based implementation is depicted in Appendix H (page 43). Note that in the zap-based implementation, Phase-2 becomes non-interactive in the BPK model, and thus the round-complexity can be reduced to five.

Next we give another way to further simplify the implementation.

Preimage-verifiable OWF based simplified implementation. We further investigate the interactions combining Phase-1 and Phase-2 of the OWF-based rZK-CKE protocol (depicted in Figure-1) when the messages $c_V^{(1)}$ and a_V are removed from Stage-2 of Phase-1 (i.e., V only sends $c_V^{(0)}$ at Stage-2 of Phase-1). The key observation here is that if the OWF f_P used by the prover is *preimage-verifiable* (as defined in Definition D.1), then such interactions can be replaced solely by letting P send a unique message $y_P = f_P(x_P)$ at the start, thereby obtaining a much more simplified 5-round implementation. In this case the proof of Theorem 4.1 remains essentially unchanged (other than being simplified). Details can be found in Appendix H (page 45).

Round-optimal rZK-CKE arguments for $\mathcal{NP}$ in the BPK model

For the above 5-round zap-based or preimage-verifiable OWF based simplified implementations, to further reduce the round-complexity, we want to fold the prover's initialization message (i.e., the first-round message that is fixed once and for all) into the third-round of the 5-round protocols (that is from the prover to the verifier). This will give 4-round (that is optimal) rZK-CKE arguments for $\mathcal{NP}$ in the BPK model. To this end, we let the verifier use OWP-based one-round perfectly-binding commitment scheme at Stage-2 of Phase-1 and replace the Blum's WIPOK protocol executed in Stage-2 of Phase-1 and Phase-3 by the Lapidot-Shamir WIPOK protocol (as in this case the verifier sends the first-round message without knowing the statement to be proved). But, the challenge here is that, for our purpose, we need a one-round OWP-based trapdoor commitment scheme to replace the two-round FSTC scheme. Specifically, we need the following cryptographic tool: A *one-round* OWP-based trapdoor commitment scheme based on DHC, in which the committer sends the one-round commitments without knowing the graph G_P (serving as $TCPK$) other than the lower and upper bounds of its size (guaranteed by the underlying $\mathcal{NP}$-reduction from $(y_P^{(0)}, y_P^{(1)})$ or y_P to G_P), and G_P is only sent in the decommitment stage after the commitment stage is finished. However, a trapdoor commitment scheme of the above described type is unknown (to the best of our knowledge). We therefore develop a trapdoor commitment of this type in this work, which is described below:

One-round commitment stage. To commit a bit 0, the committer sends a q-by-q adjacency matrix of commitments with each entry of the adjacency matrix committing to 0. To commit a bit 1,

the committer sends a q-by-q adjacency matrix of commitments such that the entries committing to 1 constitute a randomly-labelled cycle C. We remark that the underlying commitment scheme used in this stage is the one-round OWP-based perfectly-binding commitment scheme.

Two-round decommitment stage. The commitment receiver sends a Hamiltonian graph $G = (V, E)$ with size $q = |V|$ to the committer. Then, to decommit to 0, the committer sends a random permutation π, and for each non-edge of G $(i, j) \notin E$, the committer reveals the value (that is 0) that is committed to the $(\pi(i), \pi(j))$ entry of the adjacency matrix sent in the commitment stage (and the receiver checks all revealed values are 0 and the unrevealed positions in the adjacency matrix constitute a graph that is isomorphic to G via the permutation π). To decommit to 1, the committer only reveals the committed cycle (and the receiver checks that all revealed values are 1 and the revealed entries constitute a q-cycle).

The computationally-binding property of the above scheme is from the fact that the ability to decommit the same commitment-stage message both to 0 and to 1 implies extracting a Hamiltonian cycle of G (which in applications is derived from a witness to a one way function). The trapdoorness property is from the following observation: After sending a commitment to 1, one can decommit to 1 in the normal way. However, it is also possible to decommit it to 0 if one knows the Hamiltonian cycle of G. Finally, note that in the above description we have assumed the committer knows the size of the graph G sent by the commitment receiver in the decommitment stage. But it can be easily extended to the case that the committer only knows the lower-bound $l(n)$ and the upper-bound $u(n)$ of the size of G. We remark that, although the above one-round trapdoor commitment scheme is developed here to reduce round-complexity for rZK, it might be of independent value and, in particular, can be used to reduce round-complexity of other cryptographic protocols involving trapdoor commitments.

Note that in the round-optimal rZK-CKE protocols obtained thereby, we only require that the verifier uses the OWP-based one-round perfectly-binding commitment scheme. The prover, however, can still use Naor's OWF-based two-round commitment scheme. Also, note that as discussed in the proof of Theorem 4.1, the OWF and/or zap used by the prover can be only secure against standard polynomial adversaries. By simplified versions of the proof of Theorem 4.1, we get:

Corollary 5.1 *Under the existence of OWF and zap (used by the prover) that are secure against standard polynomial-time adversaries, and any OWF and OWP (used by the verifier in the key-generation phase and the protocol main-body respectively) that are secure against sub-exponential-time adversaries, any language in $\mathcal{NP}$ has a 4-round (that is optimal) rZK-CKE argument in the BPK model. (Note that the existence of (single-theorem) NIZK proofs for $\mathcal{NP}$ implies the existence of zaps.)*

Corollary 5.2 *Under the existence of any preimage-verifiable OWF (used by the prover) that is secure against standard polynomial-time adversaries, and any OWF and OWP (used by the verifier in the key-generation phase and the protocol main-body respectively) that are secure against sub-exponential-time adversaries, any language in $\mathcal{NP}$ has a 4-round (that is optimal) rZK-CKE argument in the BPK model. In particular, this implies that* round-optimal *rZK-CKE arguments for $\mathcal{NP}$ in the BPK model can be implemented with* any certified one-way permutation *(as any certified OWP is itself a preimage-verifiable OWF).*

Acknowledgments. We thank Di Crescenzo, Persiano and Visconti for many helpful and valuable discussions on earlier versions of this work. The second author would like to thank the anonymous referees of Eurocrypt'05 who pointed out the conceptual subtlety (not effective attacks) of the proof of concurrent soundness in a version of [53] submitted to Eurocrypt'05. We thank Katz and Lindell for valuable discussions, and Damgård and Dwork for kind clarifications on Σ-protocols and zaps.

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

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

[4] M. Bellare, M. Fischlin, S. Goldwasser and S. Micali. Identification protocols secure against reset attacks. In *B. Pfitzmann (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, pages 495–511. Springer-Verlag, 2001.

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

[6] M. Bellare and M. Yung. Certifying Cryptographic Tools: The Case of Trapdoor Permutations. In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1992, LNCS 740*, pages 442-460, Springer-Verlag, 1992.

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

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

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

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

[11] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 235-244, 2000. Available from: http://www.wisdom.weizmann.ac.il/∼oded/

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

[13] R. Cramer. Modular Design of Secure, yet Practical Cryptographic Protocols, PhD Thesis, University of Amsterdam, 1996.

[14] 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

[15] 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.

[16] I. Damgard. On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs. In *G. Brassard (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1989, LNCS 435*, pages 17-27. Springer-Verlag, 1989.

[17] I. Damgard. On Σ-protocols. A lecture note for the course of Cryptographic Protocol Theory at Aarhus University, 2004. Available from: http://www.daimi.au.dk/∼ivan/CPT.html

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

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

[20] G. Di Crescenzo, I. Visconti and Y. Zhao. Personal Communications, 2004.

[21] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2): 391-437, 2000. Preliminary version in *ACM Symposium on Theory of Computing*, pages 542-552, 1991.

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

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

[24] U. Feige. Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D Thesis, Weizmann Institute of Science, 1990.

[25] U. Feige, D. Lapidot and A. Shamir. Multiple Non-Interactive Zero-Knowledge Based on a Single Random String. In *IEEE Symposium on Foundations of Computer Science*, pages 308-317, 1990.

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

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

[28] 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.

[29] J. Garay and P. MacKenzie. Concurrent Oblivious Transfer. In *IEEE Symposium on Foundations of Computer Science*, pages 314-324, 2000.

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

[31] O. Goldreich, S. Goldwasser and S. Micali. How to Construct Random Functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[32] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for $\mathcal{NP}$. *Journal of Cryptology*, 9(2): 167-189, 1996.

[33] O. Goldreich, L. A. Levin and N. Nisan. On Constructing 1-1 One-Way Functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(29), 1995.

[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-Systems In *ACM Symposium on Theory of Computing*, pages 291-304, 1985.

[36] L. Guillou and J. J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In *C. G. Gnther (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1988, LNCS 330* , pages 123-128, Springer-Verlag, 1988.

[37] Y. Kalai, Y. Lindell and M. Prabhakaran. Concurrent Composition of Secure Protocols in the Timing Model. In *ACM Symposium on Theory of Computing*, pages 644-653, 2005.

[38] J. Katz. Efficient Cryptographic Protocols Preventing "Man-in-the-Middle" Attacks. Ph.D Thesis, Columbia University, 2002.

[39] J. Katz. Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 211-228. Springer-Verlag, 2003.

[40] J. Katz and R. Ostrovsky. Round-Optimal Secure Two-Party Computation. In *M. K. Franklin (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2004, LNCS 3152*, pages 335-354. Springer-Verlag, 2004.

[41] D. Lapidot and A. Shamir. Publicly-Verifiable Non-Interactive Zero-Knowledge Proofs. In *A.J. Menezes and S. A. Vanstone (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1990, LNCS 537*, pages 353-365. Springer-Verlag, 1990.

[42] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. In *ACM Symposium on Theory of Computing*, pages 683-692, 2003.

[43] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *IEEE Symposium on Foundations of Computer Science*, pages 394-403, 2003.

[44] 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.

[45] 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.

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

[47] M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 1(2): 231-262 (2004).

[48] T. Okamoto. Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1992, LNCS 740*, pages 31-53. Springer-Verlag, 1992.

[49] R. Pass. On Deniabililty in the Common Reference String and Random Oracle Models. In*D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 316-337, Springer-Verlag 2003.

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

[51] M. Yung and Y. Zhao. Constant-Round Concurrently-Secure rZK with (Real) Bare Public-Keys. Electronic Colloquium on Computational Complexity, 12(48), 2005. (This will be extended to become a version of the current submission).

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

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

[54] 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.

A Discussions on the BPK Model

A major measure of efficiency for interactive protocols is the round-complexity. Unfortunately, there are no constant-round rZK protocols in the standard model, at least for the black-box case, as implied from the works of Canetti, Killian, Petrank and Rosen [12]. To get constant-round resettable zero-knowledge protocols, the work in [11] introduced a simple model with very appealing trust requirement, the *bare public-key* (BPK) model. A protocol in BPK model simply assumes that all verifiers (whether honest or dishonest) have deposited a public key in a public file before any interaction takes place among the users. (The BPK model does allow dynamic key registrations and readers are referred to [11] for the details of dealing with dynamic key registrations.) The sole assumption is that entries in the public file were deposited before any interaction among the users takes place. But, no assumption is made on whether the public-keys deposited are unique, valid or "nonsensical" and "bad" (e.g., for which no corresponding secret-keys exist or are known) public-keys [11]. Note that an adversary may deposit many (possibly invalid or fake) public keys without any guarantee on the properties of the registered public-keys. In particular, for public-keys registered by an adversary it is *not* guaranteed that one can efficiently verify whether the adversary knows corresponding secret keys or *whether such exist* altogether. What is essentially guaranteed by the BPK model is a limitation of the number of different identities that a potential adversary may assume (note that the adversary may try to impersonate any registered user in the public-file, but it cannot act on behalf of a non-registered user), and, in fact, there are no other assurances.

The BPK model is thus very simple, and it is in fact a weaker version of the frequently used public-key infrastructure (PKI) model, which underlies any public-key cryptosystem or any digital signature scheme. Despite its apparent simplicity, the BPK model is quite powerful in achieving round-efficient cZK and rZK protocols. While cZK and rZK protocols exist both in the standard and in the BPK models [11], only in the latter case they can be constant-round, at least in the black box sense.

For cryptographic protocols in the BPK model, what kind of keys are allowed is an important *system* parameter. Thus, the complexity assumption underlying the honest users' keys is an important parameter of the protocol in the BPK model. We say a protocol is with bare *but self-certified* public-keys if it enforces public-keys to be self-certified (i.e., efficiently and publicly verifiable) for their validity and the prover interacts with (even malicious) verifiers *only* with respect to *valid* public-keys (this essentially might restrict the ability of malicious verifiers in extracting "valuable" knowledge from interactions with the honest prover). Note that the validity of a key implies more than the existence of corresponding secret-key. In particular, a valid public-key may assure some special properties (other than only an arbitrary string). Formally, the validity requirement implies that a complexity assumption supporting the validity self-certified property of the keys is needed. Recall that the initial work on BPK [11] assumed that the public key merely serves as an identity rather than a valid public-key, thus it is desired to develop protocols and protocol techniques with this property, which we informally call "real bare public keys" that formally implies potentially a weaker complexity assumption for the protocol to be secure. This is crucial for us since we attempt, among other things, to get protocols based on the weakest possible assumption.

Despite its power in achieving round-efficient cZK and rZK protocols, the soundness notion for honest verifiers with public-keys turns out to be much more complex and subtler than that of the standard model, as noted by Micali and Reyzin [44]. 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 extra gained knowledge may help this prover convincing the verifier of a *false* theorem in another interaction. Micali and Reyzin showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger: one-time, sequential, concurrent and resettable soundness. In this paper we focus on concurrent soundness which roughly means, for zero-knowledge protocols, that a malicious prover P^* cannot convince the honest verifier V of a false statement even when P^* is allowed multiple interleaving interactions with V. Micali and Reyzin also showed that any (resettable or not) black-box ZK protocols with concurrent soundness in the BPK model (for non-trivial languages outside $\mathcal{BPP}$) must run at least

four rounds [44]. It is also shown in [3, 44] that (whether resettable or not) *black-box* ZK *arguments* with resettable soundness only exist for trivial (i.e, $\mathcal{BPP}$) languages (whether in the BPK model or in standard model). Thus, it is commonly suggested that concurrent soundness might be the best one can achieve for (concurrent) verifier security of black-box (resettable) ZK arguments in the BPK model.

B Related Works and Comparisons

Constant-round (actually, round-optimal) concurrently-sound rZK arguments for $\mathcal{NP}$ in the BPK model but with self-certified public-keys were recently achieved in [18], based on a special form of PKE (with known implementation based on the decisional Diffie-Hellman DDH assumption) and length-preserving 1-1 OWF. Constant-round concurrently-sound rZK arguments for $\mathcal{NP}$ were also achieved in some stronger versions of the BPK model [45, 54].

The rZK protocol of [18] is with bare *but self-certified* public-keys, thus it is limited to certain implementations (e.g., it can be based on ElGamal but not on any RSA scheme). Thus, their technique is not suitable for further reduction of the underlying complexity assumptions. The protocol of [18] is also probably *not* knowledge-extractable even for *sequential* malicious provers, since the verifier uses perfectly-binding commitment scheme in setting the outputs of its coin-tossing sub-protocol. Indeed, the proof of concurrent soundness in [18] critically depends on the fact $x \notin L$ to reach a contradiction to the underlying hardness assumptions, and thus seemingly cannot be extended to the case $x \in L$ and deal with the issue of known witness vs. unknown one.

Constant-round sequentially-sound *concurrent* ZK (that is much weaker than rZK) arguments of knowledge in the BPK model under standard polynomial hardness assumptions were presented in a unpublished manuscript of the second author [53]. [53] presents both general OWF-based and practical DLP-based cZK arguments of knowledge in the BPK model. In particular, the DLP-based practical implementation needs only a very small constant number of exponentiations for any language that admits Σ-protocols. Actually, it is claimed in [53] that the cZK arguments of knowledge are also concurrently sound, but the proofs presented there are conceptually flawed (though, no effective attack is known). This was observed, among others, by Di Crescenzo and Visconti in a parallel work [19] (also by the authors in [52]). With the protocol structure of [53] (specifically, the protocol depicted in Figure 3, page 15 of [53]), [19] nicely fixed the flaw by running in parallel a special form of equivocal bit commitments (each bit equivocal commitment consisting of two perfectly-binding string commitments). This gives the concurrent soundness but at the cost of performance reduction to at least linear rather than constant number of exponentiations. In this work we show that both the protocol of [53] (that is sequentially-sound cZK argument of knowledge in the BPK model) and the protocol of [19] (that is concurrently-sound cZK argument of knowledge), actually all protocols having the same protocol structure of [53, 19], are *not* concurrently knowledge-extractable in the BPK model, by identifying some concurrent interleaving and malleating attacks that might be possibly of independent interest.

Constant-round non-black-box rZK protocols for $\mathcal{NP}$ in the BPK model under *standard polynomial* hardness assumptions were firstly achieved in [3] assuming collision-free hash functions, by using the non-black-box techniques developed in [1, 2]. Furthermore, they are arguments of knowledge in the BPK model in the relaxed *non-black-box* sense. Unfortunately, the (non-black-box) rZK (non-black-box) arguments of knowledge of [3] are probably not concurrently knowledge-extractable or concurrently sound in the BPK model, due to the current state of the art of the underlying non-black-box techniques used that only preserve *bounded* concurrent security.

The complication and subtleties of the notion of "argument of knowledge" in public-key models were firstly observed by Di Crescenzo and Visconti in [20]. In particular, Di Crescenzo and Visconti presented the notion of "concurrent argument of knowledge" that is a natural extension of the traditional argument of knowledge into the concurrent and public-key settings. Specifically, in the definition of "concurrent argument of knowledge" presented in [20], the knowledge-extractor is required to work in polynomial-time and is allowed to generate simulated verifier's public-key (that is, the knowledge-extractor would know the corresponding secret-key of the simulated verifier's public-key generated by itself, and may utilize the corresponding secret-key for facilitating witness extraction from concurrent malicious prover).

Di Crescenzo and Visconti also showed that concurrent argument of knowledge is strictly stronger than traditional argument of knowledge in public-key models under standard hardness assumptions. But, the construction of concurrently knowledge-extractable zero-knowledge protocols in public-key models was left over there as an open problem.

In comparison, our notion of "concurrent knowledge-extractability" is different from the notion of "concurrent argument of knowledge" of [20] in the following two ways: on one hand, the knowledge-extractor in our notion is allowed to work in super-polynomial-time (actually, sub-exponential-time). Note that black-box super-polynomial-time knowledge extraction is intrinsic in the resettable setting, as resettable (ZK or WI) protocols with black-box polynomial-time knowledge extraction are impossible for non-trivial languages outside $\mathcal{BPP}$ [11, 3]; On the other hand, the knowledge-extractor in our notion is required to use the same verifier's public-key (without knowing the corresponding secret-key) in its knowledge-extraction process. Note that, as clarified in Section 3, same public-key super-polynomial-time knowledge-extraction with complexity leveraging appears to be the only way at present to bypass the obstacle of concurrent general composition [43, 37] for achieving concurrently knowledge-extractable ZK protocols in the BPK model. Our notion of concurrent knowledge-extractability does not imply traditional argument of knowledge in public-key models, rather, we try to formalize it to be strictly stronger than concurrent soundness in public-key models under sub-exponential hardness assumptions (motivated by the concrete attacks to natural existing concurrent ZK protocols in the BPK model).

To our knowledge, the complication and subtleties of concurrent composition of proof of knowledge (POK) was first noted by Dolev, Dwork and Naor in the general context of non-malleability [21]. In [29] Garay and MacKenze noted that when POK protocols are used as building block in a larger protocol in the standard model, concurrently nested rewinding and interleaving may cause exponential blow-up of simulation time for proving the security of the larger protocol in the concurrent setting (the same problem encountered in concurrent ZK [22]). This problem was got around in [29] by working in the conditional simultaneous input model and sequentially running non-constant number of the underlying POK protocols. Concurrent *straight-line* knowledge-extraction was also investigated in the timing model [38, 39, 37], and in the random oracle model [49].

C Definitions: Concurrent Soundness, rZK and rWI in the BPK Model

In this section, we recall the definitions of concurrent soundness and resettable zero-knowledge in the BPK model (given in [11, 44, 19]). We also recall the CGGM general paradigm of [11] for achieving rWI protocols in the standard model.

Honest players in the BPK model

The BPK model consists of the following:

- F be a public-key file that is a polynomial-size collection of records (id, PK_{id}), where id is a string identifying a verifier and PK_{id} is its (alleged) public-key.
- $P(1^n, x, w, F, id, \gamma)$ be a honest prover that is a *polynomial-time* interactive machine, where 1^n is a security parameter, x is an $poly(n)$-bit string in L, w is an auxiliary input, F is a public-file, id is a verifier identity, and γ is its random-tape.
- V be a honest verifier that is a polynomial-time interactive machine working in two stages.
  1. Key generation stage. V, on a security parameter 1^n and a random-tape r, outputs a key pair (PK, SK). V then registers PK in F as its public-key while keeping the corresponding secret key SK in secret.
  2. Verification stage. V, on inputs SK, $x \in \{0,1\}^{poly(n)}$ and a random tape ρ, performs an interactive protocol with a prover and outputs "accept x" or "reject x".

The malicious concurrent prover and concurrent soundness in the BPK model

For a honest verifier V with public-key PK and secret-key SK, where (PK, SK) is the output of the key generation stage of V on a security parameter n and a random string r, an s-concurrent malicious prover P^* in the BPK model, for a positive polynomial s, is a probabilistic polynomial-time Turing machine that, on a security parameter 1^n and PK, performs an s-concurrent attack against V as follows:

P^* can perform concurrently at most $s(n)$ interactive protocols (sessions) with (the verification stage of) V; If P^* is already running $i-1$ ($1 \leq i-1 \leq s(n)$) sessions, it can select *on the fly* a common input $x_i \in \{0,1\}^{poly(n)}$ (which may be equal to x_j for $1 \leq j < i$) and initiate a new session with the verification stage of $V(SK, x_i, \rho_i)$; P^* can output a message for any running protocol, and always receive immediately the response from V (that is, P^* controls at its wish the schedule of the messages being exchanged in all the concurrent sessions). We stress that in different sessions V uses independent random-tapes in its verification stage (that is, $\rho_1, \cdots, \rho_{s(n)}$ are independent random strings).

We then say a protocol $\langle P, V\rangle$ is *concurrently sound* in the BPK model, if for any honest verifier V, for any sufficiently large n and any sufficiently-long $x \notin L$ (of length $poly(n)$) , for all positive polynomials s and all s-concurrent malicious prover P^*, the probability that V outputs "accept x" in the s-concurrent attack (i.e., in one of the $s(n)$ sessions) is negligible in n.

The malicious resetting verifier and resettable zero-knowledge in the BPK model

A malicious s-resetting malicious verifier V^*, where s is a positive polynomial, is a PPT Turing machine working in two stages so that, on input 1^n,

Stage-1. V^* receives $s(n)$ *distinct* strings $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ of equal length $poly(n)$ each, and outputs an arbitrary public-file F and a list of (without loss of generality) $s(n)$ identities $id_1, \cdots, id_{s(n)}$.

Stage-2. Starting from the final configuration of Stage-1, $s(n)$ random tapes, $\gamma_1, \cdots, \gamma_{s(n)}$, are randomly selected and then fixed for P, resulting in $s(n)^3$ deterministic prover strategies $P(x_i, id_j, \gamma_k)$, $1 \leq i, j, k \leq s(n)$. V^* is then given oracle access to these $s(n)^3$ provers, and finally outputs its "view" of the interactions (i.e., its random tapes and messages received from all its oracles).

Definition C.1 (black-box resettable zero-knowledge) *A protocol $\langle P, V\rangle$ is black-box resettable zero-knowledge for a language $L \in \mathcal{NP}$ if there exists a PPT black-box simulator S such that for every s-resetting verifier V^*, the following two probability distributions are indistinguishable. Let each distribution be indexed by a sequence of* **distinct** *common inputs $\bar{\boldsymbol{x}} = \{x_1, \cdots, x_{s(n)}\}$, $x_i \in L \cap \{0,1\}^{poly(n)}$ for $1 \leqslant i \leqslant s(n)$, and their corresponding NP-witnesses $aux(\bar{\boldsymbol{x}}) = \{w_1, \cdots, w_{s(n)}\}$:*

Distribution 1. *The output of V^* obtained from the experiment of choosing $\gamma_1, \cdots, \gamma_{s(n)}$ uniformly at random, running the first stage of V^* to obtain F, and then letting V^* interact in its second stage with the following $s(n)^3$ instances of P: $P(x_i, w_i, F, id_j, \gamma_k)$ for $1 \leq i, j, k \leq s(n)$. Note that V^* can oracle access to these $s(n)^3$ instances of P.*

Distribution 2. *The output of $S(\bar{\boldsymbol{x}})$.*

Remark. In Distribution 1 above, since V^* oracle accesses to $s(n)^3$ instances of P: $P(x_i, w_i, F, id_j, \gamma_k)$, $1 \leq i, j, k \leq s(n)$, it means that V^* may invoke and interact with the same $P(x_i, w_i, F, id_j, \gamma_k)$ multiple times, where each such interaction is called a session. We remark that there are two versions for V^* to work in Distribution 1.

1. Sequential version. In this version, a session must be terminated (either completed or aborted) before V^* initiating a new session. That is, V^* is required to terminate its current session with the current oracle $P(x_i, w_i, F, id_j, \gamma_k)$ before starting a session with any $P(x_{i'}, w_{i'}, F, id_{j'}, \gamma_{k'})$, regardless of $(i, j, k) = (i', j', k')$ or not. Thus, the activity of V^* proceeds in rounds. In each round it selects one of its oracles and conducts a terminated session with it.

2. Interleaving version. In this version the above restriction is removed and so V^* may initiate and interact, controlling the schedule of messages being exchanged, with $P(x_i, w_i, F, id_j, \gamma_k)$'s concurrently in many sessions.

However, these two versions are equivalent as shown in [11]. In other words, interleaving interactions do not help the malicious resetting verifier get more advantages on learning "knowledge" from its oracles than it can do by sequential interactions. Without loss of generality, in the rest of this paper we assume the resetting malicious verifier V^* works in the sequential version.

Definition C.2 (resettable witness indistinguishability rWI) *A protocol $\langle P, V\rangle$ is said to be resettable witness indistinguishable for an $L \in \mathcal{NP}$ if for every positive polynomial s, for every s-resetting malicious verifier V^*, two distribution ensembles of Distribution 1 (defined in Definition C.1), which are indexed by the same $\bar{\boldsymbol{x}}$ but possibly different sequences of prover's $\mathcal{NP}$-witnesses: $aux^{(1)}(\bar{x}) = \{w_1^{(1)}, \cdots, w_{s(n)}^{(1)}\}$ and $aux^{(2)}(\bar{x}) = \{w_1^{(2)}, \cdots, w_{s(n)}^{(2)}\}$, are computationally indistinguishable.*

C.1 The CGGM general paradigm for achieving rWI in the standard model

In [11] Canetti et al. presented a general paradigm for achieving rWI protocols in the standard model from any *admissible hybrid* WI protocols. A protocol is called *admissible* if the first *verifier-message*, called *"determining message"*, "essentially determines" all its subsequent messages. That is, the only freedom retained by the verifier (after sending its first message) is either to abort (or act so that the prover aborts) or to send a practically predetermined message. A typical case of admissible protocols is that the first verifier-message is a sequence of commitments that are revealed (i.e., decommitted) in subsequent verifier steps. In such a case, the verifier's freedom in subsequent steps is confined to either send an illegal decommitment (which is viewed as aborting) or properly decommit to the predetermined value. An admissible protocol is further called *hybrid* if the *prover initialization message* (in case the prover sends the first-round message of the protocol) can be fixed once and for all. Then, given an admissible hybrid protocol, i.e., in which the prover initialization message can be fixed once and for all and the first verifier message is the "determining message", the CGGM transformation transforms such a protocol into a protocol in the resettable setting by letting the prover apply a pseudorandom function on the prover initialization message, the verifier "determining message" and the common input (and verifier's public-key, in case the protocol is in the BPK model) to get the randomness to be used in the remaining computation after receiving the verifier "determining message". Furthermore, if the starting admissible hybrid protocol is WI then the transformed protocol is rWI in the standard model (or in the BPK model). For the soundness of the transformed protocol, [11] showed that if the verifier "determining message" in the starting admissible protocol is a sequence of *perfectly-hiding* commitments, then the transformed protocol constitutes a *proof* system.

The CGGM transformation directly renders us constant-round rWI *proofs* for $\mathcal{NP}$ in the standard model under the existence of two-round *perfectly-hiding* commitment schemes in which the first-round message can be fixed once and for all (e.g. the DLP or RSA based 2-round perfectly-hiding trapdoor commitment schemes presented in Section 2.1). Specifically, given any 3-round public-coin WI proof system $\langle P, V\rangle$, we transform it into an admissible hybrid WI protocol as follows: rather than sending the random-challenge after receiving the first-round message of $\langle P, V\rangle$ from the prover, the verifier first commits its random challenge on the top by running the underlying 2-round perfectly-hiding commitment scheme and later reveals the committed value as its random challenge after receiving the first-round message of $\langle P, V\rangle$ from the prover. Finally, to make the protocol resettable, we let all randomness used by the prover (other than that for generating the prover initialization message, i.e., the first-round message of the underlying 2-round perfectly-hiding commitment scheme, which however can be fixed once and for all) is got by applying a pseudorandom function on the prover initialization message, the first verifier-message (i.e., the "determining message") and the common input. The transformed rWI protocol is a *proof* system due to the *perfectly-hiding* property of the underlying perfectly-hiding commitment scheme used. But, when we replace the underlying *perfectly-*

hiding commitment scheme by a *computationally-hiding* commitment scheme, we do not know how to prove the soundness of the transformed protocol although the rWI property remains intact.

That is, from a starting 3-round public-coin WI proof system, if the CGGM transformation goes with a *computationally-hiding* (rather than perfectly-hiding) commitment scheme, then the transformed protocol is still rWI but we do not know how to prove its soundness. As a part of this work, we show that such transformed (rWI) protocol with *computationally-hiding* commitment scheme is still *computationally-sound* (i.e., it is argument rather than proof) but under sub-exponential hardness assumptions. In particular, we achieve constant-round rWI *arguments* for $\mathcal{NP}$ in the standard model under *minimal* hardness assumptions, a result unknown previously to the best of our knowledge. Specifically, we achieve 5-round (resp. 4-round) rWI arguments for $\mathcal{NP}$ in the standard model under any (sub-exponentially strong) OWF (resp. OWP).

D Major Cryptographic Tools Used

We quickly recall major cryptographic tools used in this work.

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. 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. By $[R_1; \cdots ; R_n : v]$ we denote the set of values of v that a random variable can assume, due to the distribution determined by the sequence of random processes $R_1, R_2, \cdots, R_n$. By $\Pr[R_1; \cdots ; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$.

Let $\langle P, V \rangle$ be a probabilistic interactive protocol, then the notation $(y_1, y_2) \leftarrow \langle P(x_1), V(x_2) \rangle (x)$ denotes the random process of running interactive protocol $\langle P, V \rangle$ on common input x, where P has private input x_1, V has private input x_2, y_1 is P's output and y_2 is V's output. We assume wlog that the output of both parties P and V at the end of an execution of the protocol $\langle P, V \rangle$ contains a transcript of the communication exchanged between P and V during such execution.

The security of cryptographic primitives and tools presented in this section is defined with respect to uniform polynomial-time or sub-exponential-time algorithms (equivalently, polynomial-size or sub-exponential-size circuits). When it comes to non-uniform security, we refer to non-uniform polynomial-time or sub-exponential-time algorithms (equivalently, families of circuits of polynomial or sub-exponential size).

Definition D.1 (preimage-verifiable one-way function) *A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is called a preimage-verifiable one-way function (OWF) if the following conditions hold:*

1. *Easy to compute: There exists a (deterministic) polynomial-time algorithm A such that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*

2. *Hard to invert: For every probabilistic polynomial-time PPT algorithm A', every positive polynomial $p(\cdot)$, and all sufficiently large n's, it holds $\Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$, where U_n denotes a random variable uniformly distributed over $\{0,1\}^n$. A OWF f is called* sub-exponentially strong *if for some constant c, $0 < c < 1$, for every sufficiently large n, and every circuit C of size at most 2^{n^c}, $\Pr[C(f(U_n), 1^n) \in f^{-1}(f(U_n))] < 2^{-n^c}$.*

3. *Easy to verify the preimage existence: There exists a polynomial-time computable predicate $D_f : \{0,1\}^* \longrightarrow \{0,1\}$ such that for any string y, $D_f(y) = 1$ if and only if there exists an x such that $y = f(x)$.*

We remark preimage-verifiable OWF is a generic and actually quite weak hardness assumption that includes, in particular, any certified one-way permutation and any 1-1 length-preserving one-way function. A permutation family is certified if it is easy to verify (in polynomial time) that a given function

belongs to the family. For the formal definition of certified one-way permutations, readers are referred to [6].

Below, with RSA as an example, we give more clarifications about the relationship among normal OWF, preimage-verifiable OWF and 1-1 OWF. (More detailed clarifications can be found in [33].) The difference between normal OWF and length-preserving 1-1 OWF lies in the domain of the function. Given a normal OWF from domain D to Range R, the 1-1 property of the function is defined however with respect to $\{0,1\}^* \longrightarrow \{0,1\}^*$. In other words, length-preserving 1-1 OWF is a single object, while a normal OWF could be a member of a function family. This in particular implies that any length-preserving 1-1 OWF is trivially a preimage-verifiable OWF.

For example, given a function description (N, e) where $N = pq$ for two distinct primes p and q and $gcd(e, \phi(N)) = 1$, define this generic RSA-based function (denoted by f_{GRSA}) to be $f_{GRSA}(x) = x^e \bmod N$. Then, this function f_{GRSA} (specified by (N, e)) is a normal OWF (actually OWP) from Z_N^* to Z_N^*. But, this function is NOT a preimage-verifiable OWF, because given a specific (N, e), one cannot efficiently verify whether $gcd(e, \phi(N)) = 1$ or not.

Now, consider the following *restricted* RSA-based function f_{RRSA} (specified by (N, e) such that N is a composite number and $e > N$ is a prime number). In this case, this function f_{RRSA} is a preimage-verifiable OWF from Z_N^* to Z_N^*, as the additional requirement $e > N$ ensures that $gcd(e, \phi(N)) = 1$. But, this function is clearly NOT a 1-1 function from $\{0,1\}^* \longrightarrow \{0,1\}^*$.

Definition D.2 (interactive argument system) *A pair of probabilistic polynomial-time interactive machines, $\langle P, V\rangle$, is called an interactive argument system for a language L if the following conditions hold:*

- *Completeness. For every $x \in L$, there exists a string w such that for every string z, $\Pr[\langle P(w), V(z)\rangle(x) = 1] = 1$.*
- *Soundness. For every polynomial-time interactive machine P^*, and for all sufficiently large n's and every $x \notin L$ of length n and every w and z, $\Pr[\langle P^*(w), V(z)\rangle(x) = 1]$ is negligible in n.*

An interactive protocol is called a *proof* for L, if the soundness condition holds against any (even power-unbounded) P^* (rather than only PPT P^*). An interactive system is called a public-coin system if at each round the prescribed verifier can only toss coins and send their outcome to the prover.

Definition D.3 (pseudorandom functions PRF) *On a security parameter n, let $d(\cdot)$ and $r(\cdot)$ be two positive polynomials in n. We say that*

$$\{f_s : \{0,1\}^{d(n)} \longrightarrow \{0,1\}^{r(n)}\}_{s\in\{0,1\}^n}$$

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

1. *Efficient evaluation: There exists a polynomial-time algorithm that on input s and $x \in \{0,1\}^{d(|s|)}$ returns $f_s(x)$.*
2. *Pseudorandomness: For every probabilistic polynomial-time oracle machine A, every polynomial $p(\cdot)$, and all sufficiently large n's, it holds:*

$$|\Pr[A^{F_n}(1^n) = 1] - \Pr[A^{H_n}(1^n) = 1]| < \frac{1}{p(n)}$$

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

PRFs can be constructed under any one-way function [31, 30]. The current most practical PRFs are the Naor-Reingold implementations under the factoring (Blum integers) or the decisional Diffie-Hellman hardness assumptions [47]. The computational complexity of computing the value of the Naor-Reingold functions at a given point is about two modular exponentiations and can be further reduced to only two multiple products modulo a prime (without any exponentiations!) with natural preprocessing, which is great for practices involving PRFs.

Definition D.4 (perfectly-binding string commitment scheme) *A pair of PPT interactive machines, $\langle P, V\rangle$, is called a perfectly-binding string commitment scheme, if it satisfies the following:*

Completeness. *For any security parameter n, any k (such that $k = k(n)$ for some polynomial $k(\cdot)$) and any string $s \in \{0,1\}^k$, it holds that* $\Pr[(\alpha,\beta) \leftarrow \langle P(s), V\rangle(1^n, 1^k); (t,(t,v)) \leftarrow \langle P(\alpha), V(\beta)\rangle(1^n, 1^k) : v = s]$.

Computational hiding. *For all sufficiently large n's, any PPT adversary V^* and any s, s' of equal length k (where $k = k(n)$ for some polynomial $k(\cdot)$), the following two probability distributions are computationally indistinguishable: $[(\alpha,\beta) \leftarrow \langle P(s), V^*\rangle(1^n, 1^k) : \beta]$ and $[(\alpha',\beta') \leftarrow \langle P(s'), V^*\rangle(1^n, 1^k) : \beta']$. Namely, for all sufficiently large n's, for any positive polynomial $q(\cdot)$, and every distinguishing circuit D of size $q(n)$, it holds that $|\Pr[D(1^n, 1^k, \beta) = 1] - \Pr[D(1^n, 1^k, \beta') = 1]| < \frac{1}{q(n)}$. We say the hiding property is* sub-exponentially strong *if the hiding property holds also with respect to sub-exponential-size circuits (i.e., replace the polynomial $q(\cdot)$ above by a function f of the form $f(n) = 2^{n^c}$, for some constant c, $0 < c < 1$).*

Perfect Binding. *For all sufficiently large n's, and* any *adversary P^*, the following probability is negligible in n:* $\Pr[(\alpha,\beta) \leftarrow \langle P^*, V\rangle(1^n, 1^k); (t,(t,v)) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n, 1^k);$
$(t',(t',v')) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n, 1^k) : |v| = |v'| = k \bigwedge v \neq v']$.

That is, no (even computational power unbounded) adversary P^* can decommit the same transcript of the commitment stage to two different values with non-negligible probabilities.

Below, we recall some classic perfectly-binding commitment schemes. The descriptions are referred to *bit* commitment scheme, but the extension to perfectly-binding *string* commitments is direct.

One-round perfectly-binding (computationally-hiding) commitments can be constructed based on any one-way permutation OWP [7, 34]. Loosely speaking, given a OWP f with a hard-core predict b (cf, [30]), on a security parameter n one commits a bit σ by uniformly selecting $x \in \{0,1\}^n$ and sending $(f(x), b(x) \oplus \sigma)$ as a commitment, while keeping x as the decommitment information.

Perfectly-binding commitments can also be constructed based on any one-way function but run in two rounds [46]. On a security parameter n, let $PRG : \{0,1\}^n \longrightarrow \{0,1\}^{3n}$ be a pseudorandom generator, the Naor's OWF-based two-round public-coin perfectly-binding commitment scheme works as follows: In the first round, the commitment receiver sends a random string $R \in \{0,1\}^{3n}$ to the committer. In the second round, the committer uniformly selects a string $s \in \{0,1\}^n$ at first; then to commit a bit 0 the committer sends $PRG(s)$ as the commitment; to commit a bit 1 the committer sends $PRG(s) \oplus R$ as the commitment. Note that the first-round message of Naor's commitment scheme can be fixed once and for all and, in particular, can be posted as a part of public-key in the public-key setting.

For the above perfectly-binding commitment schemes, we remark that if the underlying OWP or OWF are secure against $2^{n^{c_1}}$-time adversaries for some constant $c_1, 0 < c_1 < 1$ on a security parameter n, then the hiding property of corresponding perfectly-binding commitment schemes above also holds against $2^{n^{c_1}}$-time adversaries. Also note that the extension to perfectly-binding *string* commitments is direct.

Definition D.5 (witness indistinguishability WI) *Let $\langle P, V\rangle$ be an interactive 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 a (possibly malicious) PPT verifier V^* and the honest prover 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 $\langle P, V\rangle$ is witness indistinguishable for R_L if for every PPT interactive machine V^*, and every two sequences $W^1 = \{w^1_x\}_{x\in L}$ and $W^2 = \{w^2_x\}_{x\in L}$ for sufficiently long x, 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 by any non-uniform PPT algorithm: $\{x, view^{P(w^1_x)}_{V^*(z)}(x)\}_{x\in L,\, z\in\{0,1\}^*}$ and $\{x, view^{P(w^2_x)}_{V^*(z)}(x)\}_{x\in L,\, z\in\{0,1\}^*}$.*

Namely, for every PPT non-uniform distinguishing algorithm D, every polynomial $p(\cdot)$, all sufficiently long $x \in L$, and all $z \in \{0,1\}^*$, it holds that

$$|\Pr[D(x,z,view_{V^*(z)}^{P(w_x^1)}(x)=1] - \Pr[D(x,z,view_{V^*(z)}^{P(w_x^2)}(x)=1]| < \frac{1}{p(|x|)}$$

We say that $\langle P, V\rangle$ is sub-exponentially strong *witness indistinguishable for R_L, if for some c, $0 < c < 1$, for every sufficiently long $x \in L$ of length n, for every distinguishing circuit D of size at most 2^{n^c}, and every $z \in \{0,1\}^*$, it holds that*

$$|\Pr[D(x,z,view_{V^*(z)}^{P(w_x^1)}(x)=1] - \Pr[D(x,z,view_{V^*(z)}^{P(w_x^2)}(x)=1]| < 2^{-n^c}$$

Definition D.6 (system for proof of knowledge [5, 30]) *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 $(x,w) \in R$ all possible interactions of V with P on common input x and auxiliary input w 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 $w, r \in \{0,1\}^*$, machine K satisfies the following condition:*

  Denote by $p(x,w,r)$ the probability that the interactive machine V accepts, on input x, when interacting with the prover specified by $P^*_{x,w,r}$ (where $P^*_{x,w,r}$ denotes the strategy of P^* on common input x, auxiliary input w and random-tape r). If $p(x,w,r) > \kappa(|x|)$, then, on input x and with oracle access to $P^*_{x,w,r}$, machine K outputs a solution $w' \in R(x)$ within an expected number of steps bounded by

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

  The oracle machine K is called a knowledge extractor.

An interactive proof system $\langle P, V\rangle$ 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.

We recall two protocols of 3-round public-coin WIPOK for $\mathcal{NP}$. One is the Blum's protocol for directed Hamiltonian Cycle DHC [8] and one is the Lapidot-Shamir protocol for DHC [41].

Blum's protocol for DHC [8]. The n-parallel repetitions of Blum's basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [8] is just a 3-round public-coin WIPOK for $\mathcal{NP}$ (with knowledge error 2^{-n}) under any one-way permutation (as the first round of it involves one-round perfectly-binding commitments of a random permutation of G). But it can be easily modified into a 4-round public-coin WIPOK for $\mathcal{NP}$ under any OWF by employing Naor's two-round (public-coin) perfectly-binding commitment scheme [46]. The following is the description of Blum's *basic* protocol for DHC:

Common input. A directed graph $G = (V, E)$ with $q = |V|$ nodes.

Prover's private input. A directed Hamiltonian cycle C_G in G.

Round-1. The prover selects a random permutation, π, of the vertices V, and commits (using a perfectly-binding commitment scheme) the entries of the adjacency matrix of the resulting permutated graph. That is, it sends a q-by-q matrix of commitments so that the $(\pi(i), \pi(j))^{th}$ entry is a commitment to 1 if $(i,j) \in E$, and is a commitment to 0 otherwise.

Round-2. The verifier uniformly selects a bit $b \in \{0,1\}$ and sends it to the prover.

Round-3. If $b = 0$ then the prover sends π to the verifier along with the revealing of all commitments (and the verifier checks that the revealed graph is indeed isomorphic to G via π); If $b = 1$, the prover reveals to the verifier only the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C_G$ (and the verifier checks that all revealed values are 1 and the corresponding entries form a simple q-cycle).

We remark that the WI property of Blum's protocol for HC relies on the hiding property of the underlying perfectly-binding commitment scheme (used in its first-round). If the hiding property of the underlying perfectly-binding commitment scheme is secure against $2^{n^{c_1}}$-time adversaries for some constant $c_1, 0 < c_1 < 1$ on a security parameter n, then the WI property of Blum's protocol also holds against $2^{n^{c_1}}$-time adversaries.

The Lapidot-Shamir protocol for DHC [41]. The n-parallel repetitions of the Lapidot-Shamir basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [41] is another 3-round public-coin WIPOK for $\mathcal{NP}$ (with knowledge error 2^{-n}) under any one-way permutation (as the first round of it involves one-round perfectly-binding commitments of a random permutation of G). Again, it can be easily modified into a 4-round public-coin WIPOK for $\mathcal{NP}$ under any OWF by employing Naor's two-round (public-coin) perfectly-binding commitment scheme [46]. The following is the description of the Lapidot-Shamir *basic* protocol for DHC (that is also described in [24]):

Round-1. The prover P commits a adjacency matrix for a randomly-labelled cycle C of size q (without knowing the Hamiltonian graph to be proved). The commitment is done bit-by-bit using the one-round OWP-based perfectly-binding commitment scheme.

Round-2. The verifier V responds with a randomly chosen bit b

Round-3. Now, P is given the Hamiltonian graph $G = (V, E)$ with size $q = |V|$ to be proved and a Hamiltonian cycle C_G in G as its private input. If $b = 0$, then P opens all commitments (and V checks the revealed graph is indeed a q-cycle). If $b = 1$, then P sends a random permutation π mapping C_G (i.e., its private witness) to C (committed to its first-round message), and for each non-edge of G $(i, j) \notin E$ $(1 \leq i, j \leq q)$, P opens the value (that should be 0) committed to the $(\pi(i), \pi(j))$ entry of the adjacency matrix sent in the first-round message (and V checks all revealed values are 0 and the *unrevealed* entries in the committed adjacency matrix constitute a graph that is isomorphic to G via the permutation π).

The critical difference between Blum's protocol and the Lapidot-Shamir protocol is that, in the Lapidot-Shamir protocol the prover sends the first-round message with only the knowledge of the size of the Hamiltonian graph to be proved. Furthermore, it can be easily extended to the case that the prover knows only the lower-bound $l(n)$ and the upper-bound $u(n)$ of the size of the graph to be proved. In this case, in the first-round P commits $(u(n) - l(n) + 1)$ many adjacency matrices for $(u(n) - l(n) + 1)$ many cycles with sizes ranging from $l(n)$ to $u(n)$. In the third-round, after the size of G is clear, P only decommits with respect to the unique cycle of according size.

Again, the WI property of Blum's protocol for HC relies on the hiding property of the underlying perfectly-binding commitment scheme (used in its first-round). If the hiding property of the underlying perfectly-binding commitment scheme is secure against $2^{n^{c_1}}$-time adversaries for some constant $c_1, 0 < c_1 < 1$ on a security parameter n, then the WI property of the Lapidot-Shamir protocol also holds against $2^{n^{c_1}}$-time adversaries.

Definition D.7 (trapdoor (string) commitment scheme TC) *A (normal) trapdoor commitment scheme (TC) is a quintuple of probabilistic polynomial-time (PPT) algorithms TCGen, TCCom, TCVer, TCKeyVer and TCFake, such that*

- *Completeness.* $\forall n$, $\forall v$ *of length* k *(where* $k = k(n)$ *for some polynomial* $k(\cdot)$*),*
  $\Pr[(TCPK, TCSK) \stackrel{R}{\leftarrow} \text{TCGen}(1^n); (c, d) \stackrel{R}{\leftarrow} \text{TCCom}(1^n, 1^k, TCPK, v) :$
  $\text{TCKeyVer}(1^n, TCPK) = \text{TCVer}(1^n, 1^k, TCPK, c, v, d) = 1] = 1.$

- *Computational Binding. For all sufficiently large n's and for any PPT adversary A, the following probability is negligible in n (where $k = k(n)$ for some polynomial $k(\cdot)$):*
  $\Pr[(TCPK, TCSK) \stackrel{R}{\leftarrow} \text{TCGen}(1^n); (c, v_1, v_2, d_1, d_2) \stackrel{R}{\leftarrow} \text{A}(1^n, 1^k, TCPK) :$
  $\text{TCVer}(1^n, 1^k, TCPK, c, v_1, d_1) = \text{TCVer}(1^n, 1^k, TCPK, c, v_2, d_2) = 1 \bigwedge |v_1| = |v_2| = k \bigwedge v_1 \neq v_2]$.

- *Perfect (or Computational) Hiding. $\forall$ $TCPK$ such that* $\text{TCKeyVer}(TCPK, 1^n) = 1$ *and $\forall$ v_1, v_2 of equal length k, the following two probability distributions are identical (or computationally indistinguishable):*
  $[(c_1, d_1) \stackrel{R}{\leftarrow} \text{TCCom}(1^n, 1^k, TCPK, v_1) : c_1]$ *and* $[(c_2, d_2) \stackrel{R}{\leftarrow} \text{TCCom}(1^n, 1^k, TCPK, v_2) : c_2]$.

- *Perfect (or Computational) Trapdoorness. $\forall$ $(TCPK, TCSK) \in \{\text{TCGen}(1^n)\}$, $\exists v_1$, $\forall v_2$ such that v_1 and v_2 are of equal length k, the following two probability distributions are identical (or computationally indistinguishable):*
  $[(c_1, d_1) \stackrel{R}{\leftarrow} \text{TCCom}(1^n, 1^k, TCPK, v_1); d_2' \stackrel{R}{\leftarrow} \text{TCFake}(1^n, 1^k, TCPK, TCSK, c_1, v_1, d_1, v_2) : (c_1, d_2')]$
  and $[(c_2, d_2) \stackrel{R}{\leftarrow} \text{TCCom}(1^n, 1^k, TCPK, v_2) : (c_2, d_2)]$.

Normal trapdoor commitment schemes run in two rounds, in which the commitment receiver generates and sends $TCPK$ in the first-round.

Feige-Shamir two-round trapdoor commitments [26]. Based on Blum's protocol for DHC, Feige and Shamir developed a generic two-round (computationally-hiding and computationally-binding) trapdoor commitment scheme [26], under either any one-way permutation or any OWF (depending on the underlying perfectly-binding commitment scheme used). The $TCPK$ of the FSTC scheme (i.e., its first-round message) is $(y = f(x), G)$ (for OWF-based solution, the first-round also includes a random string R serving as the first-round message of Naor's OWF-based perfectly-binding commitment scheme), where f is a OWF and G is a graph that is reduced from y by the Cook-Levin $\mathcal{NP}$-reduction. The corresponding trapdoor is x (or equivalently, a Hamiltonian cycle in G). The following is the description of the Feige-Shamir trapdoor *bit* commitment (FSTC) scheme, in which, for our purpose, we have assumed the commitment receiver and the committer use different security parameters n and N, respectively. The extension to trapdoor *string* commitments is direct.

Round-1. Let f be a OWF, the commitment receiver randomly selects an element x of length n in the domain of f, computes $y = f(x)$, reduces y (by Cook-Levin $\mathcal{NP}$-reduction) to an instance of DHC, a graph $G = (V, E)$ with $q = |V|$ nodes, such that finding a Hamiltonian cycle in G is equivalent to finding the preimage of y. Finally, it sends (y, G) to the committer. We remark that to get OWF-based trapdoor commitments, the commitment receiver also sends a random string R of length $3N$, where N is the security parameter used by the committer.

Round-2. The committer first checks the $\mathcal{NP}$-reduction from y to G and aborts if G is not reduced from y. Otherwise, to commit 0, the committer selects a random permutation, π, of the vertices V, and commits (using the underlying perfectly-binding commitment scheme *on security parameter* N) the entries of the adjacency matrix of the resulting permutated graph. That is, it sends an q-by-q matrix of commitments so that the $(\pi(i), \pi(j))^{th}$ entry is a commitment to 1 if $(i, j) \in E$, and is a commitment to 0 otherwise; To commit 1, the committer commits an adjacency matrix containing a randomly labelled q-cycle only.

Decommitment stage. To decommit to 0, the committer sends π to the commitment receiver along with the revealing of all commitments, and the receiver checks that the revealed graph is indeed isomorphic to G via π; To decommit to 1, the committer only opens the entries of the adjacency matrix that are corresponding to the randomly labelled cycle, and the receiver checks that all revealed values are 1 and the corresponding entries form a simple q-cycle.

The (computational) trapdoorness property of the FSTC scheme is: After sending a commitment to 0 (which is indistinguishable from a commitment to 1), one can decommit to 0 in the normal way.

However, it is also possible to decommit it to 1 if one knows a Hamiltonian cycle in G. Furthermore, the distribution of a commitment to 0 *together with the "trapdoor-assistant" decommitment information to 1* is indistinguishable from the distribution of a commitment to 1 *together with the "real" decommitment information to 1* (due to the hiding property of the underlying perfectly-binding commitment scheme). This implies, by standard hybrid technique, that the distribution of commitments to 0^n together with "trapdoor-assistant" decommitment information to a random string $\hat{e}_V$ of length n is indistinguishable from the distribution of commitments to a random sting e_V of length n together with the "real" decommitment information to e_V (we will use this property in the proof of concurrent knowledge-extractability). *Again, if the hiding property of the underlying perfectly-binding commitment scheme is secure against sub-exponential-time adversaries, then both the hiding property and the trapdoorness property of the FSTC scheme hold also against sub-exponential-time adversaries.*

Witness hiding from witness indistinguishability. We recall the definition of witness hiding WH and the construction of WH from WI.

Definition D.8 (distribution of hard instances) *Let $L \in \mathcal{NP}$, and let R_L be a witness relation for L. Let $X_L \stackrel{def}{=} \{X_n\}_{n\in N}$ be a probability ensemble such that X_n ranges over $L \cap \{0,1\}^n$. We say that X_L is* hard *for R_L if for every probabilistic polynomial-time (witness-finding) algorithm F, every polynomial $p(\cdot)$, all sufficiently large n's:* $\Pr[F(X_n, 1^n) \in R_L(X_n)] < \frac{1}{p(n)}$. *We say X_L is* sub-exponentially *hard for R_L if for some constant c_L, $0 < c_L < 1$, for every sufficiently large n, and every circuit C of size at most $2^{n^{c_L}}$,* $\Pr[C(X_n, 1^n) \in R_L(X_n)] < 2^{-n^{c_L}}$. *We set c_L be 1 if X_L is* not *sub-exponentially hard.*

For example, if f is a (sub-exponentially strong) one-way function, then the probability ensemble $\{f(U_n)\}_{n\in N}$ is (sub-exponentially) hard for the witness relation $\{(f(x), x) : x \in \{0,1\}^*\}$, where U_n is uniform over $\{0,1\}^n$.

Definition D.9 (witness-hiding) *Let $L \in \mathcal{NP}$, and let R_L be a witness relation for L. Let $X = \{X_n\}_{n\in N}$ be a hard-instance ensemble for R_L. We say that a protocol $\langle P, V\rangle$ is* witness-hiding *for R_L under the instance ensemble X if for every PPT machine V^*, every polynomial $p(\cdot)$, all sufficiently large n's, and all $z \in 0, 1^*$, it holds that* $\Pr[\langle P(Y_n), V^*(z)\rangle(X_n) \in R_L(X_n)] < \frac{1}{p(n)}$, *where Y_n is arbitrarily distributed over $R_L(X_n)$.*

Let f be a (non-uniformly) one-way function, consider the probability ensemble $\{(f(U_n), f(U_n))\}_{n\in N}$ that is hard for the witness relation $R_{OR} = \{(y_0, y_1), x) : y_0 = f(x) \vee y_1 = f(x)\}$, where U_n is uniform over $\{0,1\}^n$. Then any WI protocol for the relation R_{OR} is also WH for R_{OR} under the hard-instance ensemble $\{(f(U_n), f(U_n))\}_{n\in N}$ [30].

E Concurrent Interleaving and Malleating Attacks against Zhao's Protocol and the Di Crescenzo-Visconti Protocol

We show that both the protocol of [53] (that is sequentially-sound cZK argument of knowledge in the BPK model) and the protocol of [19] (that is concurrently-sound cZK argument of knowledge), actually all protocols having the same protocol structure of [53, 19], are *not* concurrently knowledge-extractable in the BPK model, by identifying some concurrent interleaving and malleating attacks that are possibly of independent interest. This in particular shows that concurrent knowledge-extractability is strictly stronger than concurrent soundness for concurrent verifier security in public-key models when verifiers register public-keys (note that we have clarified in Section 3 that concurrent knowledge-extractability implies concurrent soundness in public-key models).

E.1 Σ-protocols and Σ_{OR}-protocols

The idea of Σ-protocols as an abstract concept is introduced by Cramer in [13]. Informally, a Σ-protocol is itself a 3-round public-coin *special* honest verifier zero-knowledge (SHVZK) protocol with special

soundness in the knowledge-extraction sense. Σ-protocols have been proved to be a very powerful cryptographic tool and are widely used in numerous important cryptographic applications including digital signatures, identification schemes, efficient electronic payment and voting systems, etc. We remark that Blum's 3-round public-coin WIPOK protocol is just a Σ-protocol for DHC. But, we note that a very large number of practical Σ-protocols also have been developed in the literature (mainly in applied cryptography). In particular, the practical Σ-protocol examples for DLP and RSA are given in below. For a good survey of Σ-protocols and their applications, readers are referred to [17, 14].

Definition E.1 *A 3-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ-protocol for a relation R if the following hold:*

- *Completeness. If P, V follow the protocol, the verifier always accepts.*
- *Special soundness. From* any *common input x of length n and any pair of accepting conversations on input x, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. Here a, e, z stand for the first, the second and the third message respectively and e is assumed to be a string of length t (that is polynomially related to n) selected uniformly at random in $\{0,1\}^t$.*
- *Special honest verifier zero-knowledge (SHVZK). There exists a probabilistic polynomial-time (PPT) simulator S, which on input x and a random challenge string e, outputs an accepting conversation of the form (a, e, z), with the probability distribution that is indistinguishable from that of the real conversation between the honest P, V on input x.*

Σ-protocol for DLP: The following is a Σ-protocol $\langle P, V\rangle$ proposed by Schnorr [50] for proving the knowledge of discrete logarithm, w, for a common input of the form (p, q, g, h) such that $h = g^w \ mod\ p$, where on a security parameter n, p is a uniformly selected n-bit prime such that $q = (p-1)/2$ is also a prime, g is an element in $\mathbf{Z}_p^*$ of order q. It is also actually the first efficient Σ-protocol proposed in the literature.

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

Σ-protocol for RSA: Let n be an RSA modulus and q be a prime. Assume we are given some element $y \in Z_n^*$, and P knows an element w such that $w^q = y \mod n$. The following protocol is a Σ-protocol (proposed in [36]) for proving the knowledge of q-th roots modulo n .

- P chooses r at random in Z_n^* and sends $a = r^q \mod n$ to V.
- V chooses a challenge e at random in Z_{2^t} and sends it to P. Here, t is fixed such that $2^t < q$.
- P sends $z = rw^e \mod n$ to V, who checks that $z^q = ay^e \mod n$, that q is a prime, that $gcd(a, n) = gcd(y, n) = 1$, and accepts iff this is the case.

The OR-proof of Σ-protocols [15]. One basic construction with Σ-protocols allows a prover to show that given two inputs x_0, x_1, it knows a w such that either $(x_0, w) \in R_0$ or $(x_1, w) \in R_1$, but without revealing which is the case. Specifically, given two Σ-protocols $\langle P_b, V_b\rangle$ for R_b, $b \in \{0,1\}$, with random challenges of, without loss of generality, the same length t, consider the following protocol $\langle P, V\rangle$, which we call Σ_{OR}. The common input of $\langle P, V\rangle$ is (x_0, x_1) and P has a private input w such that $(x_b, w) \in R_b$.

- P computes the first message a_b in $\langle P_b, V_b\rangle$, using x_b, w as private inputs. P chooses e_{1-b} at random, runs the SHVZK simulator of $\langle P_{1-b}, V_{1-b}\rangle$ on input (x_{1-b}, e_{1-b}), and 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 e and sends it to P.
- P sets $e_b = e \oplus e_{1-b}$ and computes the answer z_b to challenge e_b using (x_b, a_b, e_b, w) as input. It sends (e_0, z_0, e_1, z_1) to V.
- V checks that $e = e_0 \oplus e_1$ 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 E.1 *[15] The protocol Σ_{OR} above is a Σ-protocol for R_{OR}, where $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R_0$ or $(x_1, w) \in R_1\}$. Moreover, Σ_{OR} is witness indistinguishable proof of knowledge for R_{OR}.*

E.2 The protocol structure of [53, 19] and the difference between Zhao's protocol and the Di Crescenzo-Visconti protocol

The following is the brief protocol structure of Zhao's protocol (the protocol presented in Figure-3, page 15 of [53]).

Key-generation. Let f_V be a OWF that admits Σ-protocols. On a security parameter n, each verifier V randomly selects two elements in the domain of f_V, x_V^0 and x_V^1 of length n each, computes $y_V^0 = f_V(x_V^0)$ and $y_V^1 = f_V(x_V^1)$. V publishes (y_V^0, y_V^1) as its public-key while keeping x_V^b as its secret-key for a randomly chosen b from $\{0, 1\}$.

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

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

Phase-1. The verifier V proves to the prover P that it knows either the preimage of y_V^0 or the preimage of y_V^1, by executing the Σ_{OR}-protocol on (y_V^0, y_V^1) in which V plays the role of knowledge prover and P plays the role of knowledge verifier. Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol of this phase respectively. Here e_V is the random challenge sent by the prover to the verifier.

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

Phase-2. Let TC be a trapdoor commitment scheme with the verifier's secret-key as the trapdoor. The prover randomly selects a string $\hat{e}$, and sends $c = TC(\hat{e})$ to the verifier V.

Phase-3. Phase-3 runs essentially the underlying Σ-protocol for L but with the random challenge is set by a coin-tossing mechanism. Specifically, the prover computes and sends the first-round message of the underlying Σ-protocol, denoted a_P, to the verifier V; Then V responds with a random challenge q; Finally, P reveals $\hat{e}$ (committed in Phase-2), sets $e_P = \hat{e} \oplus q$, and computes the third-round message of the underlying Σ-protocol for L, denoted z_P, with e_P as the real random challenge.

Verifier's decision. V accepts if and only if $\hat{e}$ is decommitted correctly and $e_P = \hat{e} \oplus q$ and (a_P, e_P, z_P) is an accepting conversation for $x \in L$.

In general, Zhao's protocol can be based on any OWP, by instantiating the underlying Σ-protocols by OWP-based Blum' WIPOK for $\mathcal{NP}$ and implementing the underlying trapdoor commitment scheme by the OWP-based Feige-Shamir trapdoor commitment scheme. The concurrent ZK property of any protocol of the above protocol structure in the BPK model is from the observation that the combination of Phase-1 and Phase-2 actually constitutes an *equivocal* commitment scheme. It is also not hard to check that any protocol of the above protocol structure is sequentially sound and argument of knowledge in the BPK model. Actually, in [53] it is claimed that the above protocol is also concurrently sound, but the proofs presented there are actually conceptually flawed.

The difference between Zhao's protocol and the Di Crescenzo-Visconti protocol. The key difference between the protocol of [53] and the protocol of [19] is the underlying trapdoor commitment scheme used in Phase-2. [19] uses a special trapdoor *bit* commitment scheme that is a variant of the standard DLP-based trapdoor commitment with its decommitment information (to 0 or 1) is in turn committed to one of two perfectly-binding commitments. The augmentation of perfectly-binding commitments is crucial for fixing the proof flaw of [53] for concurrent soundness, but it also increases the computational complexity significantly, i.e., from a very small constant number of exponentiation operations to the exponentiation number that is at least linear in n, as to get trapdoor *string* commitments it is seemingly intrinsic to run the basic *bit* commitment scheme *in parallel*. In comparison, [53] uses the Damgard Σ-protocol-based trapdoor commitment scheme[16] (which is also presented in [17]). The Damgard Σ-protocol-based trapdoor commitment scheme goes as follows: The prover and the verifier first generate a hard instance (for some language that admits Σ-protocols) such that the prover does not know the corresponding witness (i.e., trapdoor); Then, to commit a value m, the prover runs the SHVZK simulator of the underlying Σ-protocol (for the generated hard-instance) on the value m to get a simulated transcript, denoted $(\hat{a}, m, \hat{z})$. Then, $\hat{a}$ is the commitment of m and $\hat{z}$ is the decommitment information. *In [53], the underlying hard instance is generated interactively (i.e. Phase-2 is interactive). But the observation here is that the hard-instance generation interactions are actually redundant there as the verifier's public-key can just be served as the underlying hard-instance.*

E.3 The concurrent interleaving and malleating attacks

The concurrent interleaving attack.

We first show a concurrent interleaving attack in which a polynomial-time malicious concurrent prover P^* can convince an honest verifier V with its public-key $PK = (y_V^0, y_V^1)$ that it "knows" the corresponding secret-key of PK with probability 1 by concurrently interacting with V in two sessions. In other words, such a attack enables P^* to personate V^*.

Specifically, on common input $PK = (y_V^0, y_V^1)$, P^* interacts with V concurrently in two sessions, and schedules the messages being exchanged in the two sessions as follows.

1. P^* interacts with V in the first session and works just as the honest prover does in Phase-1 and Phase-2. We denote by $c = TC(\hat{e})$ the Phase-2 message of the first session. When P^* moves into Phase-3 of the first session and needs to send V the first-round message, denoted by a_P, of the underlying Σ-protocol of Phase-3 of this session, P^* suspends the first session.

2. P^* initiates a second session with V; After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the second session on common input $PK = (y_V^0, y_V^1)$ (i.e. V's public-key) , P^* sets $a_P = a'_V$ and suspends the second session.

3. Now, P^* continues the execution of the first session, and sends $a_P = a'_V$ to V as the first-round message of the Σ-protocol of Phase-3 of the first session.

4. P^* runs V further in the first session. After receiving the second-round message of Phase-3 of the first session, denoted by q (i.e. the random challenge from V), P^* sets $e_P = \hat{e} \oplus q$ and suspends the first session again.

5. P^* continues the execution of the second session, and sends $e'_V = e_P = \hat{e} \oplus q$ to V as its random challenge in the second-round of the Σ_{OR}-protocol of Phase-1 of the second session. After receiving the third-round message of Phase-1 of the second session, denoted by z'_V, P^* sets $z_P = z'_V$ and suspends the second session again.

6. P^* continues the execution of the first session again, reveals $\hat{e}$ committed in Phase-2 and sends $z_P = z'_V$ to V as the last-round message of the first session.

We define a language $L_{f_V} = \{((y^0, y^1))|\exists w \; s.t. y^0 = f_V(w) \text{ OR } y^1 = f_V(w)\}$. Note that the statement "$PK \in L_{f_V}$" is always true for the *honestly generated* PK. The above concurrent interleaving attack shows that the concurrent adversary P^* can, with probability 1, convince the honest verifier with public-key PK of the statement "$PK \in L_{f_V}$", but P^* actually does *not* know the corresponding witness (i.e., the secret-key). Now, suppose f_V is secure against sub-exponential-time adversaries and Zhao's protocol or the Di Crescenzo-Visconti protocol are concurrently knowledge-extractable, then the sub-exponential-time knowledge-extractor will violate the hardness assumption of f_V, which means that, under sub-exponentially strong OWP, Zhao's protocol or the Di Crescenzo-Visconti protocol are not concurrently knowledge-extractable.

The concurrent malleating attack.

We then show another concurrent attack that enables P^* to malleate the interactions of Phase-1 of one session into a successful conversation of another concurrent session for *different* (but verifier's public-key related) statements without knowing any corresponding $\mathcal{NP}$-witnesses.

Let L' be any $\mathcal{NP}$-language admitting a Σ-protocol that is denoted by $\Sigma_{L'}$. For an honest verifier V with its public-key $PK = (y_V^0, y_V^1)$, we define a new language $L = \{(x', (y_V^0, y_V^1))|\exists w \; s.t. \; (x', w) \in R_{L'} \text{ OR } y_V^b = f_V(w) \; for \; b \in \{0, 1\}\}$. Note that for any string x' (whether $x' \in L'$ or not), the statement "$(x', (y_V^0, y_V^1)) \in L$" is always true as $PK = (y_V^0, y_V^1)$ is honestly generated. Also note that L is a language that admits Σ-protocols (as Σ_{OR}-protocol is itself a Σ-protocol). Now, we describe the concurrent malleating attack, in which P^* successfully convinces the honest verifier of the statement "$(x', (y_V^0, y_V^1)) \in L$" for *any arbitrary* $poly(n)$-bit string x' (*even* $x' \notin L'$) by concurrently interacting with V in two sessions as follows.

1. P^* interacts with V in the first session and works just as the honest prover does in Phase-1 and Phase-2. We denote by $c = TC(\hat{e})$ the Phase-2 message of the first session. When P^* moves into Phase-3 of the first session and needs to send V the first-round message, denoted by a_P, of the Σ-protocol of Phase-3 of this session *on common input* (x', y_V^0, y_V^1), P^* suspends the first session and does the following:

   - It first runs the SHVZK simulator of $\Sigma_{L'}$ (i.e., the Σ-protocol for L') on x' to get a simulated conversation, denoted by $(a_{x'}, e_{x'}, z_{x'})$, for the (*possibly false*) statement "$x' \in L$".
   - Then, P^* initiates a second session with V; After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the second session on common input (y_V^0, y_V^1) (i.e. V's public-key) , P^* sets $a_P = (a_{x'}, a'_V)$ and suspends the second session.

2. Now, P^* continues the execution of the first session, and sends $a_P = (a_{x'}, a'_V)$ to V as the first-round message of the Σ-protocol of Phase-3 of the first session.

3. P^* runs V further in the first session. After receiving the second-round message of Phase-3 of the first session, denoted by q (i.e. the random challenge from V), P^* sets $e_P = \hat{e} \oplus q$ and $e'_V = e_P \oplus e_{x'}$ and suspends the first session again.

4. P^* continues the execution of the second session, and sends $e'_V = e_P \oplus e_{x'} = \hat{e} \oplus q \oplus e_{x'}$ to V as its random challenge in the second-round of the Σ_{OR}-protocol of Phase-1 of the second session. After receiving the third-round message of Phase-1 of the second session, denoted by z'_V, P^* sets $z_P = (z_{x'}, z'_V)$ and suspends the second session again.

5. P^* continues the execution of the first session again, reveals $\hat{e}$ committed in Phase-2 and sends $z_P = (z_{x'}, z'_V)$ to V as the last-round message of the first session.

Note that $(a_{x'}, e_{x'}, z_{x'})$ is an accepting conversation for the (possibly false) statement "$x' \in L$", (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_V^0 or y_V^1, and furthermore $e_{x'} \oplus e'_V = e_P = \hat{e} \oplus q$. According to the description of Σ_{OR} (presented in Section 2.1), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation of Phase-3 of

the first-session on common input (x', y_V^0, y_V^1). That is, P^* successfully convinced V of the statement "$(x', (y_V^0, y_V^1)) \in L$" (*even* $x' \notin L'$) in the first session *but without knowing any corresponding* $\mathcal{NP}$*-witness*. We remark that such an attack is quite meaningful in certain settings. In general, the *second* attack is also similar to the attack (developed in [52]) on the Feige-Shamir ZK protocols [26] in the public-key model, but the protocol of [19] and the protocol of [26] have different protocol structures.

F Constant-Round rZK-CKE Arguments for $\mathcal{NP}$ in the BPK model under Minimal Hardness Assumption

Appendix F is an extended version of Section 4. In this section, we present constant-round (specifically, 7-round) concurrently knowledge-extractably secure rZK (rZK-CKE) arguments for $\mathcal{NP}$ with (real) bare public-keys under the minimal hardness assumption (i.e., any OWF). To this end, we also present constant-round rWI *arguments* for $\mathcal{NP}$ in the standard model *under the minimal hardness assumption*, a result unknown previously to our knowledge that is of independent value.

The high-level overview of the protocol. We first convey some ideas about the high-level overview of the protocol. Let f_V be any (*sub-exponentially strong*) OWF, each (honest) verifier V randomly selects an element x_V from the domain of f_V, and publishes $y_V = f_V(x_V)$ as its public-key with x_V as its secret-key. Let L be an $\mathcal{NP}$-language and $x \in L$ be the common input, the main-body of the protocol goes as follows: The honest prover P first generates and sends a hard-instance using a standard *polynomially-secure* OWF f_P. The hard-instance is then fixed once and for all. Then, P proves to V the existence of the preimage of the hard-instance, by executing a OWF-based resettable witness-hiding rWH protocol. After that, V proves to P that it knows either the preimage of y_V (i.e., its secret-key x_V) or the preimage of the hard-instance generated by P, by executing a OWF-based constant-round WIPOK protocol for $\mathcal{NP}$. Finally, P proves to V that it knows either a witness for $x \in L$ or the preimage of y_V (i.e., V's secret-key), by executing another OWF-based constant-round rWI argument for $\mathcal{NP}$. The detailed protocol description is depicted in Figure 1 (page 7).

The underlying complexity-leveraging. For this protocol to be provably secure, we employ the complexity-leveraging technique (that is originally introduced in [11] and also used in all previous *black-box* rZK systems in the BPK model). Specifically, the verifier V uses a security parameter N (in generating messages from it) that is also the system security parameter. But, the prover P uses a relatively smaller security parameter n (that is still polynomially related to N). The justification and discussions of the complexity-leveraging technique are given in [11]. Here, we additionally remark that letting the verifier and the prover use different security parameters is quite reasonable in the resettable setting, in which the prover is implemented by smart-cards or clients that have relatively limited computational resources and power and the verifier is normally implemented by servers that have much more computational resources and power.

Specifically, the security parameters are set as follows. On the system parameter N, suppose f_V is secure against $2^{N^{c_V}}$-time adversaries for some constant c_V, $0 < c_V < 1$. And for any $x \in L \cap \{0,1\}^{poly(N)}$, let c_L, $0 < c_L \leqslant 1$, be the constant defined in Definition 3.1. Let c be any constant such that $0 < c < \min\{c_V, c_L\}$ (in other words, $\min\{c_V, c_L\} = c + c'$ for another constant c', $0 < c' < 1$). The prover uses a relatively smaller security parameter n and uses a standard polynomially-secure OWF f_P that can be broken (brute-force wise) in time $2^{n^{c_P}}$ for some constant c_P, $c_P \geqslant 1$. Let ε be any constant such that $\varepsilon > \frac{c_P}{c}$, then we set $N = n^{\varepsilon}$. Note that N and n are polynomially related. That is, any quantity that is a polynomial of N is also (another) polynomial of n. This complexity leveraging guarantees that although any $poly(n) \cdot 2^{n^{c_P}}$-time adversary can break f_P on a security parameter n, it is still infeasible to break the one-wayness of f_V (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$). Also note that any $poly(n) \cdot 2^{n^{c_P}}$-time algorithm cannot output a witness for $x \in L$ with non-negligible probabilities, in case x is a sub-exponentially hard instance or just $x \notin L$ (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_L}}$). However, we show that for any (whether true or not) common input $x \in \{0,1\}^{poly(N)}$, if a PPT concurrent malicious P^* can convince V of the statement "$x \in L$" with non-negligible probabilities in its concurrent interactions, then there exists a black-box knowledge-extractor that, on input y_V (i.e., V's public-key), works in $poly(n) \cdot 2^{n^{c_P}}$-time and outputs a

witness for $x \in L$ also with non-negligible probabilities. Thus, under reasonable (i.e., sub-exponential) hardness assumptions on the language L, no PPT concurrent malicious prover can convince V of any (sufficiently long) statement without "knowing" a witness.

The OWF-based protocol depicted in Figure 1 (page 7) runs in 7 rounds after round combinations accordingly. In particular, the first two rounds of Phase-4 can be combined into previous phases. Now, for the protocol depicted in Figure-1, we have the following theorem:

Theorem F.1 *Under any (sub-exponentially strong) OWF, any language in $\mathcal{NP}$ has a constant-round concurrently knowledge-extractably secure rZK (rZK-CKE) argument in the BPK model.*

Proof (sketch).

The completeness of the protocol (depicted in Figure-1) is direct. Below we focus on the rZK and concurrent knowledge-extractability properties in the BPK model.

Black-box resettable zero-knowledge.

For any s-resetting adversary V^* (as defined in Appendix C), without loss of generality, we make in our analysis the following two simplifying assumptions. Firstly, we assume V^* works in the sequential version (which is equivalent to the interleaving version as discussed in Appendix C). Secondly, our analysis refers to a mental experiment in which the honest prover P utilizes a truly random function rather than a pseudorandom one. As usual, the corresponding views of the malicious verifier V^* in the two cases (i.e., random versus pseudorandom function) are computationally indistinguishable. From this point on, we identify the random-tape of P with a truly random function.

For any s-resetting adversary V^* who receives $s(N)$ *distinct* strings $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(N)}\}$, $x_i \in L \cap \{0,1\}^{poly(N)}$ for each i ($1 \leqslant i \leqslant s(N)$), and outputs an arbitrary public-file F containing $s(N)$ entries $PK_1, \cdots, PK_{s(N)}$ in its first stage, we denote by $D_t = (x_i, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, (c_V^{(0)}, c_V^{(1)}, a_V)^t)$ the "determining" message of the t-th session in which V^* interacts with its oracle $P(x_i, PK_j, \gamma_k)$, $1 \leqslant i, j, k \leqslant s(N)$ and $1 \leqslant t \leqslant (s(N))^3$. We say a public-key PK_j in F, $1 \leq j \leq s(N)$, is "covered" if the rZK simulator S has already learnt (extracted) the corresponding secret-key SK_j (if such exists).

The rZK simulation procedure is similar to (but more complicated than) that of [11]. Specifically, the rZK simulator S runs V^* as a subroutine by emulating the actions of the honest prover, and works in at most $s(N)+1$ phases such that in each phase it either successfully finishes its simulation or "covers" a new public-key in F. In each phase, S makes a simulation attempt from scratch with a new truly random function that is to be defined adaptively, and works session by session sequentially in at most $(s(N))^3$ sessions. The rZK simulator is depicted in Figure-2 (page 36), in which, for simplicity of presentation, we have assumed that V^* sends the full "determining" message $D_t = (x_i, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, (c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t)$ (rather than only $(c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t$ as specified in Figure-1) at Stage-2 of Phase-1 in the t-th session with respect to common input x_i and public-key PK_j and the honest prover instance $P(\cdot, \cdot, \gamma_k)$, $1 \leq i, j, k \leq s(N)$ and $1 \leqslant t \leqslant (s(N))^3$.

The same analysis of [11] can be directly applied here to show that the simulator S works in expected polynomial time, and the probability that S aborts with an error message due to all $n \cdot 2^n$ attempts of any inner repeat loop in its simulation have failed in extracting any knowledge (i.e., either a Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ or the expected secret-key SK_j, for any k, j, $1 \leqslant k, j \leqslant s(N)$) from V^* is negligible. Next, we show the probability that S aborts with an error message in any inner repeat loops of its simulation due to extracting a Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ (rather than the expected secret-key SK_j) is also negligible, $1 \leqslant k, j \leqslant s(N)$. The proof for this case, however, turns out to be complicated and subtle.

Specifically, we want to argue that the underlying Blum's WIPOK protocol on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ (executed in Stage-2 of Phase-1 together with Phase-3) is actually an *argument of the knowledge* of the preimage of PK_j (i.e., the secret-key SK_j). But, the subtle and complicated situation here is that before V^* finishes Phase-3, S has already proved the knowledge of the Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$

Initialization of the broken public-key set: $U \leftarrow \emptyset$

Outer Repeat Loop: Repeats up to $s(N)+1$ times

- Random function initialization: A random function f that is totally undefined.
- **Simulation from scratch:** Runs V^* on $\bar{\mathbf{x}}$ (by emulating the honest prover P) until V^* stops or S moves into Phase-4 of a session t on a common input x_i with respect to a "unbroken" public-key $PK_j \notin U$. We denote by $D_t = (x_i, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, (c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t)$ the "determining" message of the t-th session, $1 \leq i, j, k \leq s(N)$ and $1 \leqslant t \leqslant (s(N))^3$. Note that $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ is generated by S itself, emulating the actions of the honest prover in Stage-1 of Phase-1 of *all* sessions with respect to honest prover instances $P(\cdot,\cdot,\gamma_k)$. Note also that D_t may be equal to $D_{\hat{t}}$ for some $\hat{t}$, $1 \leq \hat{t} \leq t$.

  During this process, in each session t' with respect to a *distinct* "determining" message $D_{t'} = (x_{i'}, F, (j', PK_{j'}), (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^{k'}, (c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^{t'})$ and a "*covered*" public-key $PK_{j'}$, $1 \leq i', j', k' \leq s(N)$ and $1 \leqslant t' \leqslant (s(N))^3$, then S uses independent random coins in its remaining computation after receiving the "determining" message by defining f on the new point $D_{t'}$ (note that the $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^{k'}$ is fixed once and for all), and uses the (assumed known) secret-key $SK_{j'}$ as its witness in Phase-4. For any session with a "determining" message that is identical to that of some previous session, then S just copies what sent in that previous session.
- If V^* stops then S also stops and outputs the transcript up to now.
- If S moves into Phase-4 of the session t with respect to the uncovered public-key PK_j, then it implies that V^* has successfully finished Phase-3. We denote by $e_{V^*}^t$, $z_{V^*}^t$, the first-round message and the second-round message of Phase-3 in the t-th session respectively. Note that $((c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t, e_{V^*}^t, z_{V^*}^t)$ constitute a successful conversation of the Blum's WI protocol on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$. let T denote the partial transcript that $V^*(T) = D_t$ and for any prefix T' of T $V^*(T') \neq D_t$. That is, T is the transcript on which V^* will send the underlying "determining" message D_t *at the first time*. In other words, T determines the smallest $\hat{t}$, $1 \leq \hat{t} \leq t$, such that $D_t = D_{\hat{t}}$. Then S rewinds V^* to T and does the following:
- **Inner Repeat Loop:** Repeats up to $n \cdot 2^n$ times
  - Redefines (the output of) f on the point $D_{\hat{t}} (= D_t)$ to be a new independent random string (of according length), which in particular includes a new random challenge, denoted $\hat{e}_{V^*}^{\hat{t}}$, to be sent as the first-round message of Phase-3 of any session (from T) with respect to the same "determining" message $D_{\hat{t}} (= D_t)$.
  - Runs V^* from T.
  - Whenever S moves into Phase-4 of a session again with the same "determining" message $D_{\hat{t}} (= D_t)$, and furthermore this is the *first* session *from* T that S moves into Phase-4 with respect to a *uncovered* public-key, it means that S receives again a valid second-round message of Phase-3, denoted $\hat{z}_{V^*}^{\hat{t}}$, with respect to $\hat{e}_{V^*}^{\hat{t}}$ and the same $(c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t (= (c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^{\hat{t}})$ such that $((c_{V^*}^{(0)}, c_{V^*}^{(1)}, a_{V^*})^t, \hat{e}_{V^*}^{\hat{t}}, \hat{z}_{V^*}^{\hat{t}})$ constitute another successful conversation of the Blum's WI protocol on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$, from which S can efficiently extract either SK_j or a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$. If the extracted knowledge is SK_j, then S sets $U \leftarrow U \cup \{j\}$ and goes to **Outer Repeat Loop**. Otherwise (i.e., the extracted knowledge is a Hamiltonian Cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$), then S aborts with an error message.
  - In any other cases, S proceeds to the next iteration of **Inner Repeat Loop**.
- End [of inner repeat loop]
- In case all the above $n \cdot 2^n$ attempts have failed, S aborts with an error message

End [of outer repeat loop]

Figure 2. The black-box rZK simulator

in Phase-2. Note that the $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ is fixed once and for all (which can be viewed as the public-key of the honest prover instance $P(\cdot,\cdot,\gamma_k)$), and furthermore V^* is resettingly (more than concurrently) interacting with the honest prover instances. As demonstrated in Section 3 and Appendix E, normal argument of knowledge and even concurrent soundness do not guarantee knowledge-extractability in such setting. In particular, one may argue that, by rewinding the honest prover instances arbitrarily, V^* may potentially forge the interactions on $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ provided by the honest prover in Phase-2 of one session into successful but "false" interactions on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ in Stage-2 of Phase-1 and Phase-3 of another session with respect to public-key PK_j, in the sense that although the interactions are valid but V^* actually does not know the corresponding secret-key SK_j. This means

that, in such case the interactions on $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$ executed in Phase-3 together with Stage-2 of Phase-1 are not any longer an *argument of the knowledge* of the preimage of PK_j, although it is always a system for proof of knowledge of either SK_j or a Hamiltonian cycle of $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$. What save us here is the concurrent (not resettable) WI property of the Blum's protocol for HC.

Below, we construct an algorithm $\hat{S}$ that emulates the real rZK simulator while *concurrently* (not resettingly) running the Blum's protocol for HC. That is, on common inputs $\{(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^1, \cdots, (y_P^{(0)}, y_P^{(1)}, G_P, R_P)^{s(N)}\}$ $\hat{S}$ concurrently interacts with $s(N)$ instances of the knowledge prover, denoted $\hat{P}$, of Blum's protocol for HC by playing the role of knowledge verifier. We denote each of the $s(N)$ instances of $\hat{P}$ by $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$, $1 \leqslant k \leqslant s(N)$; At the same time, $\hat{S}$ runs the s-resetting malicious V^* as a subroutine by playing the role of the honest prover, and sends $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ as the Stage-1 message of Phase-1 whenever V^* initiates a session with the honest prover instance $P(\cdot, \cdot, \gamma_k)$. $\hat{S}$ emulates the rZK simulator S but with the following modification: whenever $\hat{S}$ needs to send a "fresh" first-round message of Blum's protocol for HC on $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ in Phase-2 with respect to a "determining" message (this happens due to either V^* sends a distinct "determining" message in one session or $\hat{S}$ needs rewinding V^* and redefining the underlying random function f to extract knowledge used by V^* in a successful execution of Stage-2 of Phase-1 and Phase-3 with respect to an uncovered public-key), it initiates a new session with $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$, and forwards the first-round message received from $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$ to V^*. Then, $\hat{S}$ runs V^* further, and in case V^* successfully reveals the assumed challenge (that is *perfectly-bindingly* committed to the underlying "determining" message in question) then $\hat{S}$ returns back the revealed challenge to $\hat{P}$ as its own challenge in the according simultaneous session of Blum's protocol for HC, and returns back the third-round message received from $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$ to V^*. For a session with a "determining" message that is identical to that of some previous sessions, $\hat{S}$ just copies what sent in the previous sessions. Note that in this case, $\hat{S}$ may still possibly need to interact with $\hat{P}$ in some *existing* concurrent session to get some third-round message (in case V did not reveal or invalidly revealed the random challenge perfectly-bindingly committed to the underlying "determining" message in all previous sessions but correctly reveals it in the current session). But, the key point here is that in this case S does not need to initiate a new concurrent session with $\hat{P}$.

Note that from the viewpoint of V^*, the behavior of $\hat{S}$ is identical to the behavior of the real rZK simulator, where the real rZK simulator S generates $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$'s and provides the corresponding Phase-2 messages by itself (rather than get by interacting with the knowledge prover instances $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$'s of the Blum's protocol for HC). The key observation here is that although V^* is actually resettingly interacting with $\hat{S}$, but $\hat{S}$ only concurrently interacts with the instances of $\hat{P}$ and never rewinds $\hat{P}$. The underlying reason is just that, in any session Phase-2 interactions take place only after V^* sent the "determining" message at Stage-2 of Phase-1 that determines the subsequent behaviors of V^* in that session. Note that in this case, the (concurrent) WI property of the Blum's protocol for HC on common input $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ actually implies witness hiding (WH), which means no PPT algorithm can output a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ even by concurrently interacting with $\hat{P}((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k)$. Also note that on common input $((y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k, PK_j)$, Phase-3 together with Stage-2 of Phase-1 is always a system for proof of knowledge of either a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ or the preimage of PK_j (i.e., SK_j), which means that with overwhelming probabilities $\hat{S}$ (or the real rZK simulator S) always can extract either a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ or the corresponding secret-key SK_j within time inversely proportional to the probability that V^* successfully finishes Phase-3 (by rewinding V^* and redefining the underlying random function). But, the WH property of Blum's protocol for HC shows that with overwhelming probabilities, $\hat{S}$ (or the real rZK simulator S) never outputs a Hamiltonian cycle in $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)^k$ in its simulation that is done in expected polynomial-time. Here, a subtle point needs to be further addressed. Specifically, the normal WH property is defined with respect to probabilistic (strict) polynomial-time algorithms, but here $\hat{S}$ works in expected polynomial-time. But, by Markov's inequality, it is easy to see that if the WH

property of a protocol holds with respect to any probabilistic (strict) polynomial-time algorithms, then it also holds with respect to any expected polynomial-time algorithms. (Specifically, if an algorithm A breaks the WH property of a protocol with non-negligible probability $\frac{1}{q(n)}$ in expected $p(n)$-time, where $q(\cdot)$ and $p(\cdot)$ are positive polynomials, then we can construct another strict polynomial-time A' that runs A $p(n) \cdot (2 \cdot q(n))$ steps. If A outputs nothing in the $p(n) \cdot (2 \cdot q(n))$ steps, then A' aborts; otherwise, A' outputs whatever A outputs. By Markov's inequality, it's easy to see that A' breaks the WH property of the underlying protocol with probability at least $\frac{1}{2 \cdot q(n)}$ that is also non-negligible. So, if a protocol is WH with respect to any probabilistic (strict) polynomial-time algorithms, then it is also WH with respect to any expected polynomial-time algorithms.)

Finally, conditioned on the rZK simulator S does not abort with error messages, the indistinguishability between the simulated transcript (outputted by S) and the real interaction transcript (between V^* and honest prover instances) is from the rWI property of Phase-4 combined with Phase-1. Actually, the combination of Phase-4 and Phase-1 constitutes a protocol for $\mathcal{NP}$ in the standard model that holds the rWI property, as shown by the CGGM general paradigm for achieving rWI protocols in the standard model from admissible hybrid WI protocols [11]. But, as discussed in Appendix C.1, the soundness property of the protocol combining Phase-4 and Phase-1 is not direct, because the underlying trapdoor commitment scheme used in Phase-1 is only computationally (rather than perfectly) hiding. Actually, as we shall see, the proof for the soundness property in this case, and especially for concurrent soundness and concurrent knowledge-extractability of the *whole* protocol in the BPK model, turns out to be much more complicated and subtle, which is to be elaborated next.

Comments: Note that in the proof of rZK, we require nothing about the public-keys registered by V^* in F. What we need in the simulation is the POK property of Blum's protocol executed in Stage-2 of Phase-1 and Phase-3, which does hold with respect to *any* common input (in particular, *any* public-key registered by V^*, whether valid or not). That is, our protocol is with what we informally call real bare public-keys. Also note that for the OWF f_P used by the prover, it can be only secure against standard polynomial-time adversaries, as the one-wayness of f_P is only used to guarantee the computationally-binding property of the underlying FSTC scheme against malicious polynomial-time verifiers (in proving black-box resettable zero-knowledge).

Black-box concurrent knowledge-extractability.

We show that for any (whether true or not) common input $x \in \{0,1\}^{poly(N)}$, if a PPT s-concurrent malicious P^* can convince an honest verifier V (with public-key PK and secret-key SK) of the statement "$x \in L$" with non-negligible probability p_x in one of the $s(N)$ concurrent interactions, then there exists a black-box knowledge-extractor E that, on input PK with oracle accessing P^*, works in $poly(n) \cdot 2^{n^{c_P}}$-time and outputs a witness for $x \in L$ also with non-negligible probabilities. Note that according to the underlying complexity leveraging on the security parameters N and n, no $poly(n) \cdot 2^{n^{c_P}}$-time algorithm can break the one-wayness of f_V used by V in forming its public-key on security parameter N (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$). Also note that any $poly(n) \cdot 2^{n^{c_P}}$-time algorithm cannot output a witness for $x \in L$ with non-negligible probabilities, in case x is a sub-exponentially hard instance or just $x \notin L$ (because $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_L}}$).

Taking PK as its input, E randomly chooses j from $\{1, \cdots, s(N)\}$ and runs P^* as a subroutine by playing the role of the honest verifier with public-key PK. Note that E does not know the corresponding secret-key SK. In each session t, $1 \leqslant t \leqslant s(N)$, after receiving the Stage-1 message of Phase-1, denoted $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^t, G_{P^*}^t, R_{P^*}^t)$, E first checks whether or not $G_{P^*}^t$ is $\mathcal{NP}$-reduced from $(y_{P^*}^{(0)}, y_{P^*}^{(1)})^t$ and $R_{P^*}^t$ is of length $3N$. If the checking is successful, then E tries to find a Hamiltonian cycle in $G_{P^*}^t$ by brute-force searching in $2^{n^{c_P}}$-time.

- If E finds a Hamiltonian cycle in $G_{P^*}^t$, then E sets the Stage-2 message of Phase-1 of the t-th session, denoted $((c_V^{(0)})^t, (c_V^{(1)})^t, a_V^t)$, as follows: it randomly chooses one random string $(e_V^{(0)})^t$ from $\{0,1\}^n$, computes $(c_V^{(0)})^t = Com(1^N, R_{P^*}^t, (e_V^{(0)})^t)$ by using the underlying Naor's perfectly-binding commitment scheme Com, and computes $(c_V^{(1)})^t = TCCom(1^N, (G_{P^*}^t, R_{P^*}^t), 0^n)$ by using the underlying Feige-Shamir trapdoor commitment scheme (note that, $(c_V^{(1)})^t$ commits to

0^n rather than a random string in $\{0,1\}^n$ as the honest verifier does). Then, on common input $(((y_P^{(0)}, y_P^{(1)})^t, G_{P^*}^t, R_{P^*}^t), PK)$ V computes the first-round message, denoted a_V^t, of (n-parallel repetitions of) Blum's WIPOK for $\mathcal{NP}$ for showing the knowledge of either SK or a Hamiltonian cycle in $G_{P^*}^t$. *Note that the first-round message of Blum's WIPOK for $\mathcal{NP}$ is computed without using any witness knowledge (i.e., either SK or a Hamiltonian cycle in $G_{P^*}^t$)*; In case P^* successfully finishes Phase-2 of the t-th session, E moves into Phase-3. After receiving the first-round message of Phase-3 of the t-th session, denoted e_V^t, E computes the second-round message of Phase-3, denoted z_V^t (i.e., the third-round message of Blum's WIPOK for showing the knowledge of either SK or a Hamiltonian cycle in $G_{P^*}^t$), *by using the extracted Hamiltonian cycle in $G_{P^*}^t$ as its witness*; Finally, in Phase-4 of the t-th session, E decommits $(c_V^{(1)})^t$ to a random string $(e_V^{(1)})^t$ of length n, *by using the extracted Hamiltonian cycle in $G_{P^*}^t$ as the trapdoor.*

- If there exists *no* Hamiltonian cycle in $G_{P^*}^t$, then E sets and sends the Stage-2 message of Phase-1 of the t-th session, i.e., $((c_V^{(0)})^t, (c_V^{(1)})^t, a_V^t)$, just as above. But, whenever P^* successfully finishes Phase-2 of the t-th session and sends to E the first-round message of Phase-3 of the t-th session (i.e., e_V^t), E aborts with an error message (as it has no witness for generating the next message).

In the j-th session with respect to a common input x_j selected by P^* on the fly, we denote by $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^j, G_{P^*}^j, R_{P^*}^j)$ the Stage-1 message of Phase-1 of the j-th session, and by $((c_V^{(0)})^j, (c_V^{(1)})^j, a_V^j)$ the Stage-2 message of Phase-1 of the j-th session that is set as above specified (where $(c_V^{(1)})^j = TCCom(1^N, (G_{P^*}^j, R_{P^*}^j), 0^n)$). In case P successfully finishes the j-th session (i.e., Phase-4 of the j-th session), we denote by $a_{P^*}^j$ the first-round message of Phase-4 of the j-th session, by $(e_V^{(1)})^j$ the random challenge (of length n) sent by E in the second-round of Phase-4 (by decommitting $(c_V^{(1)})^j$ with the extracted Hamiltonian cycle in $G_{P^*}^j$ as the trapdoor), and by $z_{P^*}^j$ the third-round message of Phase-4 of the j-th session, where $(a_{P^*}^j, (e_V^{(1)})^j, z_{P^*}^j)$ constitutes a successful conversation of (n-parallel repetitions of) Blum's WIPOK for showing the knowledge either a witness for $x_j \in L$ or the corresponding secret-key SK. Then, after receiving the last-round message (i.e., $z_{P^*}^j$), E rewinds P^* to the state that it just sent $a_{P^*}^j$, decommits $(c_V^{(1)})^j$ to a different random string $(e_V^{(1)})^{j'}$ (that is taken uniformly from $\{0,1\}^n / \{(e_V^{(1)})^j\}$)) by using the extracted Hamiltonian cycle in $G_{P^*}^j$ as the trapdoor, and runs P^* further expecting to receive another valid third-round message, denoted $z_{P^*}^{j'}$, of Phase-4 of the j-th session.

For any x, denote by q_x the probability that P^* successfully convinces $E(PK)$ of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions (without rewinding in the j-th session). Then, with probability about $\frac{(q_x)^2}{s(N)}$ E will output either a witness w for $x \in L$ or the corresponding secret-key SK such that $PK = f_V(SK)$ in $poly(n) \cdot 2^{n^{c_P}}$-time, by rewinding the j-th session for a randomly chosen j from $\{1, \cdots, s(N)\}$. As $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$ and f_V is secure against any circuit of size $2^{N^{c_V}}$, we know with probability negligibly close to $\frac{(q_x)^2}{s(N)}$ E will output a witness w for $x \in L$. Now, to establish the concurrent knowledge-extractability property, all left is to show that $|p_x - q_x|$ is negligible *for any* x, where p_x is the probability that P^* successfully convinces the honest verifier with public-key PK of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions. This is done by establishing a series of hybrid experiments.

We first consider a mental experiment in which P^* concurrently interacts with an imaginary verifier $\widehat{V}$ with the same public-key PK and secret-key SK. $\widehat{V}$ mimics the real honest verifier V with public-key PK and secret-key SK but with the following modifications: For any session t, $1 \leqslant t \leqslant s(N)$, in case P^* successfully finishes Phase-2 and sends to $\widehat{V}$ the first-round message of Phase-3, $\widehat{V}$ enumerates all possible Hamiltonian cycles of $G_{P^*}^t$ by brute-force searching in $2^{n^{c_P}}$-time, where $((y_{P^*}^{(0)}, y_{P^*}^{(1)})^t, G_{P^*}^t, R_{P^*}^t)$ is the Stage-1 message of Phase-1 of the t-th session. If there exists *no* Hamiltonian cycle in $G_{P^*}^t$, then $\widehat{V}$ aborts with an error message (*although it can continue the execution with SK as its witness*).

For any x, denote by $\hat{p}_x$ the probability that P^* successfully convinces the imaginary verifier $\widehat{V}$ with public-key PK of the statement "$x \in L$" in one of the $s(N)$ concurrent sessions. We want to show that for any x $|p_x - \hat{p}_x|$ is negligible in n. Note that the only difference between the interactions between

P^* and $\widehat{V}$ and the interactions between P^* and the real honest verifier V is that: for any session t, $1 \leqslant t \leqslant s(N)$, the real honest verifier always continues the execution of Phase-3 by using SK as its witness in forming the second-round message of Phase-3, in case P^* successfully finished Phase-2 and sent the first-round message of Phase-3 ; but $\widehat{V}$ may abort in this case if it finds the $G^t_{P^*}$ is "false" (i.e. there exists no Hamiltonian cycle in $G^t_{P^*}$) by brute-force searching in $2^{n^{c_P}}$-time. Thus, the fact that for any x $|p_x - \hat{p}_x|$ is negligible is from the following lemma.

Lemma F.1 *For all positive polynomials $s(\cdot)$ and all s-concurrent malicious P^*, the probability that there exists a t, $1 \leqslant t \leqslant s(N)$, such that P^* can successfully finish Phase-2 with respect to a false $G^t_{P^*}$ (i.e., $G^t_{P^*}$ contains no Hamiltonian cycle) in the t-th session of the $s(N)$ concurrent sessions (against the real honest verifier V with public-key PK) is negligible in n. Note that any quantity that is negligible in n is also negligible in N.*

Proof (of Lemma F.1). We show that if a PPT s-concurrent adversary P^* can convince V (with public-key PK) of a false $G^t_{P^*}$ with non-negligible probability $p'(n)$ in Phase-2 of one of the $s(N)$ concurrent sessions, then this will violate the hiding property of the underlying *perfectly-binding* commitment scheme, denoted Com, used by V in Phase-1 that is run on security parameter N. Note that according to the hiding property of the underlying perfectly-binding commitment scheme Com, given two strings $\hat{e}_0$ and $\hat{e}_1$ that are taken uniformly at random from $\{0,1\}^n$ and $C = Com(1^N, R^t_{P^*}, \hat{e}_b)$ for a randomly chosen bit $b \in \{0,1\}$, no $2^{N^{c_V}}$-time (non-uniform) algorithm can distinguish whether C commits to $\hat{e}_0$ or to $\hat{e}_1$ (i.e., guess the bit b correctly) with non-negligible advantage over $1/2$, even with $\hat{e}_0$, $\hat{e}_1$ and the secret-key of V (i.e., SK) as its non-uniform inputs.

We construct a (non-uniform) algorithm A who takes $(1^n, (\hat{e}_0, \hat{e}_1, SK), C)$ as input and wants to guess b with a non-negligible advantage over $1/2$, where $\hat{e}_0$ and $\hat{e}_1$ are taken uniformly at random from $\{0,1\}^n$ and $C = Com(1^N, R_{P^*}, \hat{e}_b)$ for a randomly chosen bit $b \in \{0,1\}$. E randomly selects j from $\{1, \cdots, s(N)\}$, runs P^* as a subroutine by playing the role of the honest verifier V with secret-key SK in any session other than the j-th session. In the j-th session, after receiving $G^j_{P^*}$ from P^* at Stage-1 of Phase-1, E first checks whether there exists a Hamiltonian cycle in $G^j_{P^*}$ or not by brute-force searching in time $2^{n^{c_P}}$. If E finds a Hamiltonian cycle in $G^j_{P^*}$, then E randomly guesses the bit b and stops. Otherwise (i.e., there exists no Hamiltonian cycle in $G^j_{P^*}$), E runs P^* further and continues the interactions of the j-th session as follows: E gives C to P^* as the assumed commitment to $(e^{(0)}_V)^j$ at Stage-2 of Phase-1. After receiving the first-round message of Phase-2 (i.e., the first-round of Blum's protocol for proving the existence of a Hamiltonian cycle in $G^j_{P^*}$) that contains n committed adjacency matrices, E first opens all the committed adjacency matrices by brute-force in $poly(n) \cdot 2^{n^{c_P}}$-time (note that E can do this as the underlying perfectly-binding commitment scheme used by the prover in forming these n committed adjacency matrices is run on security parameter n). For each revealed graph G^j_k $(1 \leq k \leq n)$ (described by the corresponding opened adjacency matrix entries) , we say G^j_k is a 0-valid graph if it is isomorphic to $G^j_{P^*}$, or a 1-valid graph if it contains a Hamiltonian cycle of the same size of $G^j_{P^*}$. We say the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_b$-valid ($b \in \{0,1\}$) if for all k, $1 \leq k \leq n$, G^j_k is a $\hat{e}^{(k)}_b$-valid graph, where $\hat{e}^{(k)}_b$ denotes the k-th bit of $\hat{e}_b$. Note that for the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$, E can determine whether it is $\hat{e}_0$-valid or $\hat{e}_1$-valid in time $poly(n) \cdot 2^{n^{c_P}}$. Then, E outputs 0 if the set $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_0$-valid but not $\hat{e}_1$-valid. Similarly, E outputs 1 if the set $\{G^j_1, \cdots, G^j_n\}$ is $\hat{e}_1$-valid but not $\hat{e}_0$-valid. In other cases, E just randomly guesses the bit b.

The key observation here is that if $G^j_{P^*}$ is false (i.e., containing no Hamiltonian cycle), then for each revealed graph it cannot be both a 0-valid graph and a 1-valid graph. Similarly, for false $G^j_{P^*}$, the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ cannot be both $\hat{e}_0$-valid and $\hat{e}_1$-valid for different $\hat{e}_0 \neq \hat{e}_1$. Furthermore, suppose C commits to $\hat{e}_b$ ($b \in \{0,1\}$), then for false $G^j_{P^*}$ with probability $1 - 2^{-n}$ the set of revealed graphs $\{G^j_1, \cdots, G^j_n\}$ is not $\hat{e}_{1-b}$-valid (as $\hat{e}_{1-b}$ is taken uniformly at random from $\{0,1\}^n$). As the value j is randomly chosen from $\{1, \cdots, s(N)\}$, we conclude that E can successfully guess the bit b with probability at least $(1-2^{-n}) \cdot \frac{p'(n)}{s(N)} + \frac{1}{2}(1 - \frac{p'(n)}{s(N)}) = \frac{1}{2} + \frac{1}{2} \cdot \frac{p'(n)}{s(N)} - 2^{-n} \cdot \frac{p'(n)}{s(N)}$ in time $poly(n) \cdot 2^{n^{c_P}}$. That is, E successfully guesses the bit b with non-negligible advantage over $1/2$ in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$,

which violates the hiding property of the underlying perfectly-binding commitment scheme Com used by V that is run on the security parameter N. □

Thus, we have established that for any x $|p_x - \hat{p}_x|$ is negligible. To show that for any x $|p_x - q_x|$ is negligible, we can show that for any x $|\hat{p}_x - q_x|$ is negligible. This is done by establishing another hybrid experiment.

Specifically, we consider another hybrid experiment, in which a PPT algorithm $\widehat{E}$ takes (PK, SK) as its input (that is, $\hat{E}$ takes both the verifier's public-key and the corresponding secret-key as its input), and runs P^* as a subroutine by mimicking the knowledge-extractor E (who only takes PK as input) but with the following modification: For any session t, $1 \leqslant t \leqslant s(N)$, in case P^* successfully finishes Phase-2 and sends to $\widehat{E}$ the first-round message of Phase-3, $\widehat{E}$ enumerates all possible Hamiltonian cycles of $G^t_{P^*}$ by brute-force searching in $2^{n^{c_P}}$-time, where $((y^{(0)}_{P^*}, y^{(1)}_{P^*})^t, G^t_{P^*}, R^t_{P^*})$ is the Stage-1 message of Phase-1 of the t-th session. If there *exists* a Hamiltonian cycle in $G^t_{P^*}$, then $\widehat{E}$ continues the execution by forming the second-round message of Phase-3 of the t-th session (for showing the knowledge of either SK or a Hamiltonian cycle of $G^t_{P^*}$) *but using SK as its witness just as the real honest verifier does* (note that in this case E continues the execution with the extracted Hamiltonian cycle of $G^t_{P^*}$ as the corresponding witness). If there exists *no* Hamiltonian cycle in $G^t_{P^*}$, then $\widehat{E}$ aborts with an error message just as E (or $\widehat{V}$) does (*although in this case $\widehat{E}$ can continue the execution with SK as its witness*).

Note that the difference between the interactions between P^* and the imaginary verifier $\widehat{V}$ in the first hybrid experiment and the interactions between P^* and $\widehat{E}$ is that: in any session t, $1 \leqslant t \leqslant s(N)$, of the interactions between P^* and $\widehat{V}$, $\widehat{V}$ always commits (and accordingly decommits to) a random string of length n (i.e., $(e^{(1)}_V)^t$) by using the underlying FSTC scheme (just as the honest verifier V does), but in the interactions between P^* and $\widehat{E}$, $\widehat{E}$ always commits 0^n and then decommits to a random string of length n by using the brute-force extracted Hamiltonian cycle of $G^t_{P^*}$ as the trapdoor (just as E does). The difference between the interactions between P^* and $\widehat{E}$ and the interactions between P^* and E is that: E always uses the brute-force extracted Hamiltonian cycle of $G^t_{P^*}$ as its witness in Phase-3 of any session t, $1 \leqslant t \leqslant s(N)$, but $\widehat{E}$ always uses the verifier's secret-key SK as its witness (just as the honest verifier does).

For any x, denote by $\hat{q}_x$ the probability that P^* can convince $\widehat{E}$ of the statement "$x \in L$" in one of the $s(N)$ sessions. Then if there exists an x such that $|\hat{p}_x - \hat{q}_x|$ is non-negligible, we can break the hiding and trapdoorness properties of the underlying FSTC scheme in the following way: We construct a (non-uniform) algorithm $\widehat{A}$ that takes (x, PK, SK) as the (non-uniform) input and runs P^* as a subroutine by emulating either $\widehat{V}$ or $\widehat{E}$ (Note that either $\widehat{V}$ or $\widehat{E}$ can be emulated in $poly(n) \cdot 2^{n^{c_P}}$-time). Whenever $\widehat{A}$ finds P^* successfully convinces the statement "$x \in L$" $\widehat{A}$ outputs 1, otherwise $\widehat{A}$ outputs 0. Clearly, by standard hybrid technique, if $|\hat{p}_x - \hat{q}_x|$ is non-negligible we can break the hiding and trapdoorness properties of the underlying FSTC scheme in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$. Similarly, we can also prove that for any x $|\hat{q}_x - q_x|$ is negligible, as otherwise we can break the WI property of Blum's protocol for $\mathcal{NP}$ in time $poly(n) \cdot 2^{n^{c_P}} \ll 2^{N^{c_V}}$. Thus, we get that for any x $|\hat{p}_x - q_x|$ is negligible, and so is $|p_x - q_x|$ for any x because we have shown for any x $|p_x - \hat{p}_x|$ is negligible. This finishes the proof for concurrent knowledge-extractability of the protocol depicted in Figure-1 in the BPK model. □

G Constant-Round rWI Arguments for $\mathcal{NP}$ under Minimal Hardness Assumptions in the Standard Model

The proof of Theorem 4.1 implies that the protocol depicted in Figure-1 contains a sub-protocol, specifically the combination of Phase-1 and Phase-2, that is constant-round rWI *argument* for $\mathcal{NP}$ *in the standard model* under minimal hardness assumptions, a result unknown previously that might be possibly of independent interest. (We remark that the construction of the rWI argument is actually conceptually simple, but as we have seen, the security analyses are however complicated and subtle.) The OWF-based rWI argument is reproduced in Figure 3 (page 42).

Common input. A directed graph $G = (V, E)$ with $|V| = n$.

P **private input.** A Hamiltonian cycle C_G in G, a random string R_P of length $3N$ that serves as the first-round message of Naor's OWF-based perfectly-binding commitment scheme Com used by V, and a random string γ that is an n-bit string serving as the randomness seed of a PRF.

Complexity-leverage used. Let c_V, $0 < c_V < 1$, be the constant that the hiding property of the underlying perfectly-binding commitment scheme Com used by the verifier holds against any circuit of size $2^{N^{c_V}}$. The prover also uses the Naor's OWF-based perfectly-binding commitment scheme that is, however, run on the security parameter n and thus can be brute-force decommitted in 2^{c_P}-time for a constant c_P, $c_P \geqslant 1$. Then we set $\epsilon > c_P/c_V$ and $N = n^{\epsilon}$. This ensures that one can open the committed adjacency matrices (sent by prover in the first-round of the underlying Blum's protocol for HC), or enumerate all potential Hamiltonian cycles of G in time $poly(n) \cdot 2^{n^{c_P}}$, which is still lesser than the time it would take to break the hiding property of the underlying perfectly-binding commitment scheme used by the verifier that is run on security parameter N (because $2^n < 2^{N^{c_1}}$).

Phase-1. Phase-1 contains of two stages:

Stage-1. P sends the random string R_P of length $3N$ to V (that serves as the first-round message of Naor's OWF-based perfectly-binding commitment scheme Com used by V). *Note that R_P is fixed once and for all.*

Stage-2. V randomly chooses a string e_V from $\{0,1\}^n$, computes $c_V = Com(1^N, R_P, e_V)$ by using the underlying Naor's perfectly-binding commitment scheme Com that is run on the security parameter N. V also randomly selects a string R_V of length $3n$ that serves as the first-round message of Naor's perfectly-binding commitment scheme used by P (that is run on the security parameter n). Finally, V sends (c_V, R_V) to P. *From then on, all randomness used by P in the remaining computation is got by applying $PRF(\gamma, \cdot)$ on the "determining" message (G, c_V, R_V).*

Phase-2. P proves to V the existence of a Hamiltonian cycle in G by executing the (n-parallel repetitions of) Blum's protocol for HC, in which V sends the second-round message (i.e., the assumed random challenge) by just revealing e_V committed to c_V. Note the first-round message of Phase-2 (from P to V) consists of n committed adjacency matrices computed by running the underlying perfectly-binding commitment scheme *on security parameter n.*

Figure 3. Constant-round rWI arguments for $\mathcal{NP}$ in the standard model under minimal hardness assumptions

The OWF-based protocol depicted in Figure-3 runs in 5 rounds, but the round-complexity can be trivially reduced to four by employing OWP-based one-round perfectly-binding commitment scheme. The computational soundness of the protocol is direct from the proof of Lemma 4.1, and the rWI property is direct from the general CGGM paradigm for achieving rWI protocols in the standard model from admissible hybrid WI systems (as described in Appendix C.1). Thus, we have the following corollary:

Corollary G.1 *Under any (sub-exponentially strong) OWF (resp. OWP), any language in $\mathcal{NP}$ has a 5-round (resp. 4-round) rWI argument in the standard model.*

H Simplified and Round-Optimal Implementations

Appendix H is an extended version of Section 5. In this section, we investigate various simplified implementations of rZK-CKE arguments in the BPK model with reduced round-complexity but under stronger (still quite general) hardness assumptions. We also show how to achieve round-optimal rZK-CKE arguments for $\mathcal{NP}$ in the BPK model under still quite general hardness assumptions. The simplified and round-optimal implementations involve novel uses of a number of cryptographic tools, together with introducing a new tool, specifically, a OWP-based *one-round* trapdoor commitment scheme. (Actually, the results presented in this section are what originally appear in [51], but with result presentation order is reversed. Readers can refer to [51] for more details.)

Zap-based simplified implementation. A natural way to simplify the implementation of the protocol depicted in Figure-1, while remaining almost the same protocol structure, is to replace Blum's WI protocol (with commitment of the random challenge, i.e., $e_V^{(0)}$, on the top) executed in Phase-1 and Phase-2 by *zap* developed in [23]. Zap is itself a 2-round public-coin WI *proof* for $\mathcal{NP}$. It can be easily modified into a 2-round rWI *proof* for $\mathcal{NP}$, by applying a PRF on the first-round message, denoted ξ,

Key generation. Let f_V be any OWF that is secure against $2^{N^{c_V}}$-time adversaries. On a security parameter N, each honest verifier V randomly selects an element x_V of length N, computes $y_V = f_V(x_V)$; V also randomly selects a string, denoted ξ, of length $l(n)$ that serves as the first-round message of the underlying zap, where $l(\cdot)$ is a polynomial. Then, V posts (y_V, ξ) as its public-key PK while keeping x_V as its secret-key SK. (If P uses Naor's OWF-based perfectly-binding commitment scheme (that is run on security parameter n), V also deposits a random string R_V of length $3n$ as a part of its public-key serving as the first-round message of Naor's commitment scheme used by P.)

Common input. An element $x \in L \cap \{0,1\}^{poly(N)}$ (let c_L, $0 < c_L \leqslant 1$, be the constant defined in Definition 3.1.), the public-file F and an index j that specifies the j-th entry of F, i.e., $PK_j = (y_V^{(j)}, \xi^{(j)}, R_V^{(j)})$. Note that the system security parameter is N.

P **private input.** An $\mathcal{NP}$-witness w for $x \in L$, a pair of random strings (γ_1, γ_2), where γ_1 is a $poly(n)$-bit string and γ_2 is an n-bit string serving as the randomness seed of a PRF.

V **private input.** Private key SK_j. For simplicity of presentation, except explicitly clarified we denote $PK_j = f_V(SK_j)$.

Complexity-leverage used. Let c_V, $0 < c_V < 1$, be the constant that the one-wayness of the OWF f_V, and thus the hiding property of the underlying perfectly-binding commitment scheme used by the verifier all hold against any circuit of size $2^{N^{c_V}}$ (which in turn guarantees that the WI property of the underlying WI protocol for $\mathcal{NP}$ executed in Stage-2 of Phase-1 and Phase-3, the hiding and trapdoorness properties of the underlying trapdoor commitment scheme all hold against any circuit of size $2^{N^{c_V}}$). The prover uses a relatively smaller security parameter n. Let c_P, $c_P \geqslant 1$, be the constant that: for all sufficiently large n's, the size of G_P (reduced from $(y_P^{(0)}, y_P^{(1)})$) is bounded by n^{c_P}. Let c be any constant such that $0 < c < \min\{c_V, c_L\}$ and ε be any constant such that $\varepsilon > \frac{c_P}{c}$, then we set $N = n^{\varepsilon}$.

Phase-1. Phase-1 consists of two stages:

Stage-1. Let f_P be any (polynomially-secure) OWF. On security parameter n, P randomly selects two elements $x_P^{(0)}$ and $x_P^{(1)}$ of length n each in the domain of f_P, computes $y_P^{(b)} = f_P(x_P^{(b)})$ for $b \in \{0,1\}$, reduces $(y_P^{(0)}, y_P^{(1)})$ to a directed graph G_P by Cook-Levin $\mathcal{NP}$-reduction such that finding a Hamiltonian cycle in G_P is equivalent to finding the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$. If V uses Naor's OWF-based perfectly-binding commitment scheme (that is run on security parameter N), P also randomly selects a string R_P of length $3N$ serving as the first-round message of Naor's OWF-based perfectly-binding commitment scheme used by V. Finally, P sends $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)$ to V. The randomness used by P in this process is γ_1, *which means that the* $(y_P^{(0)}, y_P^{(1)}, G_P, R_P)$ *is fixed once and for all.*

Stage-2. V first checks whether or not G_P is reduced from $(y_P^{(0)}, y_P^{(1)})$ and R_P is of length $3N$. If the checking is successful, V randomly chooses one string e_V from $\{0,1\}^n$, computes $c_V = TCCom(1^N, (G_P, R_P), e_V)$ by using the underlying trapdoor commitment scheme. Then, on common input $((y_P^{(0)}, y_P^{(1)}, G_P, R_P), PK_j)$ V computes the first-round message, denoted a_V, of (n-parallel repetitions of) Blum's WIPOK for $\mathcal{NP}$ for showing the knowledge of either SK_j or a Hamiltonian cycle in G_P (equivalently, the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$). Finally, V sends (c_V, a_V) to P. *From then on, all randomness used by* P *in the remaining computation is got by applying* $PRF(\gamma_2, \cdot)$ *on the "determining" message* $D = (x, F, (j, PK_j), (y_P^{(0)}, y_P^{(1)}, G_P, R_P), (c_V, a_V))$.

Phase-2. On common input G_P and with $\xi^{(j)}$ as the first-round message of the underlying zap, P computes and sends the second-round message, denoted Π, of the underlying zap for showing the existence of a Hamiltonian cycle in G_P (equivalently, the preimage of either $y_P^{(0)}$ or $y_P^{(1)}$).

Phase-3. V and P continue the WIPOK protocol for $\mathcal{NP}$ suspended at Stage-2 of Phase-1. If V successfully convinces P of the knowledge of either SK_j or a Hamiltonian cycle in G_P, then goto Phase-4. Otherwise, P aborts. We denote by e_V, z_V, the first-round message and the second-round message of Phase-3 respectively.

Phase-4. P proves to V that it "knows" either the witness w for $x \in L$ or the secret-key SK_j, by executing the (n-parallel repetitions of) Blum's protocol for $\mathcal{NP}$ on common input (x, PK_j), in which V sends the assumed random challenge by just revealing e_V committed to c_V.

Figure 4. Constant-round rZK-CKE arguments for $\mathcal{NP}$ with zaps

and the common input to get the randomness for generating the second-round message, denoted Π [23]. Furthermore, the first-round message of a zap can be fixed once and for all, and thus can be posted as a part of the underlying public-key. The zap-based implementation is depicted in Figure-4 (page 43).

Note that in the zap-based implementation, Phase-2 becomes non-interactive in the BPK model, and thus the round-complexity can be reduced to five. Specifically, Phase-2, Phase-3 and Phase-4 can

be combined into three rounds (in other words, Phase-2 and Phase-3 can be combined into Phase-4). We remark that the fact that zap is a *proof* (rather than argument) system much eases the proof of concurrent knowledge-extractability, as the "proof" property trivially guarantees the truth of Lemma 4.1. Also note that, as discussed in the proof of Theorem 4.1, the OWF f_P and the underlying zap used by the prover can be only secure against standard polynomial-time adversaries. By a simplified version of the proof of Theorem 4.1, we get:

Corollary H.1 *Under the existence of OWF and zap (used by the prover) that are secure against standard polynomial-time adversaries, and any OWF used by the verifier that is secure against sub-exponential-size circuits, any language in $\mathcal{NP}$ has a 5-round concurrently knowledge-extractably secure rZK (rZK-CKE) argument in the BPK model. Note that the existence of (single-theorem) NIZK proofs for $\mathcal{NP}$ implies the existence of zaps.*

Preimage-verifiable OWF based implementation and its practical instantiations. We further reinvestigate the combination of Phase-1 and Phase-2 of the OWF-based rZK-CKE protocol (depicted in Figure-1) when the messages $c_V^{(1)}$ and a_V are removed from Stage-2 of Phase-1 (i.e., V only sends $c_V^{(0)}$ at Stage-2 of Phase-1). Such interactions play (*and only play*) the following dual roles in the proof of Theorem 4.1:

One-wayness in the proof of rZK. That is, such interactions do not render non-negligible advantages to any s-resetting malicious verifier V^* in extracting a Hamiltonian cycle in G_P (equivalently, the preimage of either $y_P^{(0)}$ or $y_P^{(0)}$) sent at Stage-1 of Phase-1 that is fixed once and for all. The one-wayness property plays a critical role in the proof of rZK to ensure the rZK simulator can extract the corresponding secret-key within time inversely proportional to the probability that V^* successfully finishes the execution of Phase-3 of a session with respect to a "uncovered" public-key.

Preimage verifiability in the proof of concurrent knowledge-extractability. That is, for any s-concurrent malicious prover P^*, if such interactions are successfully finished in the t-th session, $1 \leqslant t \leqslant s(N)$, then with overwhelming probabilities there exists a Hamiltonian cycle in G_P^t (equivalently, a preimage of $(y_P^{(0)}, y_P^{(1)})^t$). We remark that the preimage-verifiability property plays a critical role in the proof of concurrent knowledge-extractability, as otherwise the malicious P^* can distinguish whether it is interacting with honest verifier instances or with the knowledge extractor.

The key observation here is that if the OWF f_P used by the prover (in forming the Stage-1 message of Phase-1) is *preimage-verifiable* (as defined in Definition D.1), then such interactions of above in the OWF-based protocol can be replaced by only letting P send a unique message $y_P = f_P(x_P)$ on the top, with the proof of Theorem 4.1 remains essentially unchanged (other than much more simplified). This brings us (much more) simplified 5-round *preimage-verifiable* OWF based rZK-CKE arguments for $\mathcal{NP}$ in the BPK model, which is reproduced in Figure-5 (page 45). Again, as discussed in the proof of Theorem 4.1, the preimage-verifiable OWF f_P used by the prover can be only secure against standard polynomial-time adversaries. By a simplified version of the proof of Theorem 4.1, we get:

Corollary H.2 *Under any preimage-verifiable OWF (used by the prover) that is secure against standard polynomial-time adversaries and any OWF (used by the verifier) that is secure against sub-exponential-time adversaries, any language in $\mathcal{NP}$ has a 5-round concurrently knowledge-extractably secure rZK (rZK-CKE) argument in the BPK model. Note that preimage-verifiable OWF is a quite general hardness assumption, which includes, in particular, any certified OWP and any 1-1 length-preserving OWF.*

Key generation. Let f_V be any OWF that is secure against $2^{N^{c_V}}$-time adversaries. On a security parameter N, each honest verifier V randomly selects an element x_V of length N, computes $y_V = f_V(x_V)$, publishes y_V as its public-key PK while keeping x_V as its secret-key SK. (If P uses Naor's OWF-based perfectly-binding commitment scheme in Phase-2, V also deposits a random string R_V of length $3n$ as a part of its public-key serving as the first-round message of Naor's commitment scheme.)

Common input. An element $x \in L \cap \{0,1\}^{poly(N)}$ (let c_L, $0 < c_L \leqslant 1$, be the constant defined in Definition 3.1.), the public-file F and an index j that specifies the j-th entry of F, i.e., $PK_j = (y_V^{(j)}, R_V^{(j)})$. Note that the system security parameter is N.

P **private input.** An $\mathcal{NP}$-witness w for $x \in L$, a pair of random strings (γ_1, γ_2), where γ_1 is a $poly(n)$-bit string and γ_2 is an n-bit string serving as the randomness seed of a PRF.

V **private input.** Private key SK_j. For simplicity of presentation, except explicitly clarified we denote $PK_j = f_V(SK_j)$.

Complexity-leverage used. Let c_V, $0 < c_V < 1$, be the constant that the one-wayness of the OWF f_V, and thus the hiding property of the underlying perfectly-binding commitment scheme used by the verifier all hold against any circuit of size $2^{N^{c_V}}$ (which in turn guarantees that the WI property of the underlying WI protocol for $\mathcal{NP}$ executed in Stage-2 of Phase-1 and Phase-3, the hiding and trapdoorness properties of the underlying trapdoor commitment scheme all hold against any circuit of size $2^{N^{c_V}}$). The prover uses a relatively smaller security parameter n. Let c_P, $c_P \geqslant 1$, be the constant that: for all sufficiently large n's, the size of G_P (reduced from y_P) is bounded by n^{c_P}. Let c be any constant such that $0 < c < \min\{c_V, c_L\}$ and ε be any constant such that $\varepsilon > \frac{c_P}{c}$, then we set $N = n^{\varepsilon}$.

Phase-1. Phase-1 consists of two stages:

Stage-1. Let f_P be any *preimage-verifiable* OWF. On security parameter n, P randomly selects an element x_P of length n in the domain of f_P, computes $y_P = f_P(x_P)$, reduces y_P to a directed graph G_P by Cook-Levin $\mathcal{NP}$-reduction such that finding a Hamiltonian cycle in G_P is equivalent to finding a preimage of y_P. If V uses Naor's OWF-based perfectly-binding commitment scheme (that is run on security parameter N), P also randomly selects a string R_P of length $3N$ serving as the first-round message of Naor's OWF-based perfectly-binding commitment scheme used by V. Finally, P sends (y_P, G_P, R_P) to V. The randomness used by P in this process is γ_1, *which means that the* (y_P, G_P, R_P) *is fixed once and for all.*

Stage-2. V first checks the validity of (y_P, G_P) (i.e., whether or not G_P is reduced from y_P *and the preimage of* y_P *exists*) and aborts if it is not valid. If it is valid, V randomly chooses a string e_V from $\{0,1\}^n$ and computes $c_V = FSTCCom(1^N, (G_P, R_P), e_V)$ (that is, V commits e_V using the underlying Feige-Shamir trapdoor commitment FSTC scheme). V sends c_V to P, and proves to P that it knows either the preimage of PK_j (i.e., SK_j) or a Hamiltonian cycle in G_P (equivalently, a preimage of y_P), by executing the (n-parallel repetitions of) Blum's WIPOK for $\mathcal{NP}$ on common input $((G_P, R_P), PK_j)$ in which V plays the role of knowledge prover and P plays the role of knowledge verifier. Denote by a_V the first-round message (that is from knowledge prover V to knowledge verifier P) of the n-parallel repetitions of Blum's WIPOK protocol, *then all randomness used by* P *(from then on after receiving* (c_V, a_V)*) is got by applying* $PRF(\gamma_2, \cdot)$ *on the "determining" message* $D = (x, F, (j, PK_j), (y_P, G_P, R_P), (c_V, a_V))$. If V successfully finishes the Blum's WIPOK protocol in this stage and P accepts, then goto Phase-2. Otherwise, P aborts.

Phase-2. P proves to V that it knows either the witness w for $x \in L$ or the secret-key SK_j, by executing the (n-parallel repetitions of) Blum's protocol for $\mathcal{NP}$ on common input (x, PK_j), in which V sends the assumed random challenge by just revealing e_V committed to c_V.

Figure 5. Constant-round rZK-CKE arguments for $\mathcal{NP}$ with preimage-verifiable OWF

H.1 Round-optimal rZK-CKE arguments for $\mathcal{NP}$ in the BPK model

For the 5-round zap-based or preimage-verifiable OWF based rZK-CKE protocols described above, if the verifier V uses the OWP-based one-round perfectly-binding commitment scheme then the prover only needs to send $(y_P^{(0)}, y_P^{(1)}, G_P)$ or (y_P, G_P) in the first-round. To further reduce the round-complexity, we want to combine $(y_P^{(0)}, y_P^{(1)}, G_P)$ (or (y_P, G_P)) into the third-round of the 5-round protocol (that is from the prover to the verifier), thereby obtaining 4-round (that is optimal) rZK-CKE arguments for $\mathcal{NP}$. Recall that $(y_P^{(0)}, y_P^{(1)}, G_P)$ (or (y_P, G_P)) is used by V in two ways: On one hand, it forms the $\mathcal{NP}$-statement (to be precise, a directed graph reduced from (G_P, PK_j) by $\mathcal{NP}$-reduction) to be proved by V by Blum's WIPOK for HC at Stage-2 of Phase-1; On the other hand, it serves $TCPK$ of the underlying FSTC scheme with a Hamiltonian cycle of G_P as the trapdoor $TCSK$. To combine

$(y_P^{(0)}, y_P^{(1)}, G_P)$ (or (y_P, G_P)) into the third-round while remaining the same protocol structure, we need the following cryptographic tools.

1. A 3-round OWP-based WIPOK for HC, in which the prover sends the first-round message without knowing the $\mathcal{NP}$-statement (i.e., a directed graph) to be proved, other than the lower and upper bounds of the size of the graph (guaranteed by the underlying $\mathcal{NP}$-reduction).

2. A *one-round* OWP-based trapdoor commitment scheme based on HC, in which the committer sends the one-round commitments without knowing the graph G_P (serving as $TCPK$) other than the lower and upper bounds of its size (guaranteed by the underlying $\mathcal{NP}$-reduction from $(y_P^{(0)}, y_P^{(1)})$ or y_P to G_P), and G_P is only sent in the decommitment stage after the commitment stage is finished.

For the first needed cryptographic tool of above, we note that the Lapidot-Shamir OWP-based 3-round WIPOK for HC [41] (described in Appendix D) is just the protocol of the type we need.

Thus, the challenge here is to develop a one-round trapdoor commitment scheme of the above described type, which however, to our knowledge, is unknown in the literature previously. To our purpose, we develop the trapdoor commitment of such type in this work, which is described below:

One-round commitment stage. To commit a bit 0, the committer sends a q-by-q adjacency matrix of commitments with each entry of the adjacency matrix committing to 0. To commit a bit 1, the committer sends a q-by-q adjacency matrix of commitments such that the entries committing to 1 constitute a randomly-labelled cycle C. We remark that the underlying commitment scheme used in this stage is the one-round OWP-based perfectly-binding commitment scheme.

Two-round decommitment stage. The commitment receiver sends a Hamiltonian graph $G = (V, E)$ with size $q = |V|$ to the committer. Then, to decommit to 0, the committer sends a random permutation π, and for each non-edge of G $(i, j) \notin E$, the committer reveals the value (that is 0) that is committed to the $(\pi(i), \pi(j))$ entry of the adjacency matrix sent in the commitment stage (and the receiver checks all revealed values are 0 and the unrevealed positions in the adjacency matrix constitute a graph that is isomorphic to G via the permutation π). To decommit to 1, the committer only reveals the committed cycle (and the receiver checks that all revealed values are 1 and the revealed entries constitute a q-cycle).

The computationally-hiding property of the above scheme is directly from that of the underlying perfectly-binding commitment scheme. The computationally-binding property of the above scheme is from the fact that the ability to decommit the same commitment-stage message both to 0 and to 1 implies extracting a Hamiltonian cycle of G. The trapdoorness property is from the following observation: After sending a commitment to 1, one can decommit to 1 in the normal way. However, it is also possible to decommit it to 0 if one knows the Hamiltonian cycle of G. Finally, note that in the above description we have assumed the committer knows the size of the graph G sent by the commitment receiver in the decommitment stage. But it can be easily extended to the case that the committer only knows the lower-bound $l(n)$ and the upper-bound $u(n)$ of the size of G. In this case, in commitment stage P sends $(u(n) - l(n) + 1)$ many adjacency matrices with vertex-sizes ranging from $l(n)$ to $u(n)$. In the decommitment stage, after the size of G is clear, P only decommits with respect to the unique adjacency matrix of according size. We remark that, although the above one-round trapdoor commitment scheme is developed here to reduce round-complexity for rZK, it might be of independent value and, in particular, can be used to reduce round-complexity of other cryptographic protocols involving trapdoor commitments.

After showing the existence of the cryptographic tools needed for obtaining round-optimal rZK-CKE arguments for $\mathcal{NP}$ with real bare public-keys, we remark that in the round-optimal rZK-CKE protocols obtained thereby we only require the verifier use the OWP-based one-round perfectly-binding commitment scheme. The prover, however, can still use Naor's OWF-based two-round commitment scheme. Also note that the existence of zaps should actually be a different assumption than the assumption of

preimage-verifiable OWFs, and it might be hard to say one is weaker than another. Thus, we have the following corollaries:

Corollary H.3 *Under the existence of OWF and zap (used by the prover) that are secure against standard polynomial-time adversaries, and any OWF and OWP (used by the verifier in the key-generation phase and the protocol main-body respectively) that are secure against sub-exponential-time adversaries, any language in $\mathcal{NP}$ has a 4-round (that is optimal) rZK-CKE argument with real bare public-keys in the BPK model.*

Corollary H.4 *Under the existence of any preimage-verifiable OWF (used by the prover) that is secure against standard polynomial-time adversaries, and any OWF and OWP (used by the verifier in the key-generation phase and the protocol main-body respectively) that are secure against sub-exponential-time adversaries, any language in $\mathcal{NP}$ has a 4-round (that is optimal) rZK-CKE argument. In particular, this implies that* round-optimal *rZK-CKE arguments for $\mathcal{NP}$ with real bare public-keys can be implemented with* any certified one-way permutation *(as any certified OWP is itself a preimage-verifiable OWF).*

ECCC ISSN 1433-8092
http://eccc.hpi-web.de/