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

Constant-Round Concurrently-Secure rZK in the (Real) Bare Public-Key Model

Moti Yung [*] Yunlei Zhao [†]

Abstract

We present constant-round concurrently secure (sound) resettable zero-knowledge (rZK-CS) arguments in the bare public-key (BPK) model. Our constructions deal with general NP ZK-arguments as well as with highly efficient ZK-arguments for number-theoretic languages, most relevant to identification scenarios. These are the first constant-round protocols of this type in the original *real* BPK model, where nothing is assured about public-keys generated by malicious verifiers. As part of this work, we develop a one-round trapdoor commitment scheme that is based on any one-way permutation, which is of independent interest and, in particular, can be used to reduce the round-complexity of other cryptographic protocols involving trapdoor commitments.

1 Introduction

Zero-knowledge (ZK) protocols are remarkable since they allow a prover to validate theorems to a verifier without giving away any other knowledge (computational advantage). This notion was suggested in [21] and its generality was demonstrated in [20] and since its introduction ZK has found numerous and extremely useful applications. ZK protocols can be executed in many environments (models), and resettable zero-knowledge (rZK) is the most restrictive model of zero-knowledge to date. It was put forth by Canetti, Goldreich, Goldwasser and Micali [6], motivated by implementing zero-knowledge provers using smart-cards or other devices that may be (maliciously) reset to their initial conditions and/or can not 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 introduced by Dwork, Naor and Sahai [14].) In a nutshell, an rZK protocol remains secure even if the verifier concurrently interacts with the prover polynomially many times, each time restarting an interaction with the prover using the same configuration and random tape.

ZK protocols for general languages (i.e., NP) show important plausibility, since many important statements are in NP. In addition, ZK has many direct efficient applications (mainly employing number theoretic statements). A major direct application is identification (ID) schemes advocated in [18, 16]. While in many settings the paradigm transform ZK protocols to ID schemes, these protocols fail to secure whenever the prover is resettable.

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 work of Canetti, Killian, Petrank and Rosen [7, 8]. To get constant-round resettable zero-knowledge protocols [6] introduced a simple model with very appealing trust requirement, the *bare public-key* (BPK) model. A protocol in BPK model simply assumes that all players (whether honest or dishonest) have deposited a public key in a public file before any interaction takes place among the

[2]Department of Computer Science, Columbia University, New York, NY, USA. moti@cs.columbia.edu
[3]Hewlett-Packard Laboratories, Filton Road, Bristol BS34 8QZ, UK. 990314@fudan.edu.cn

ISSN 1433-8092

users[*]. 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), further, 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 digital signature scheme. In the PKI case, a secure association between a key and its owner is crucial, while in the BPK case no such association is required. The BPK model is also a weaker version of the frequently used *certified public-key model*, where the validity of any public-keys (even malicious chosen) can be *efficiently* verified.

Despite its power in achieving round-efficient rZK protocols, the soundness notion in the BPK model turns out to be much more subtle and complex than in the standard model, as noted by Micali and Reyzin [25]. In public-key models, a verifier V has a secret key SK, corresponding to its public-key PK. A malicious prover P^* could potentially gain some knowledge about SK from an interaction with the verifier. This gained knowledge might help it 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 (black-box) rZK protocols with concurrent soundness in the BPK model must run at least four rounds [25].

It has been a major open problem to achieve constant-round concurrently sound rZK (rZK-CS) arguments for $\mathcal{NP}$, especially with optimal round-complexity, in the BPK model. Due to its hardness, a recent result by [13], introduced a solution in the *stronger* public key model (namely, the *certified public-key model*). The security proof of [13] critically depends on the requirement that the validity of public-keys registered by even malicious verifiers can be efficiently verified (this is despite the title of that work). rZK-CS arguments for $\mathcal{NP}$ with optimal round-complexity in other stronger versions of the BPK model were also achieved in [26, 31]. But, achieving constant-round rZK-CS arguments, especially with optimal round-complexity, in the *real* BPK model is still open. We explore this basic open problem in this work.

We further note that in spite of its significance to practice, especially to smart-card based identification and other potential validations, all previous rZK proof systems concentrated on plausibility. The inefficiency in previous rZK systems is in part due to their basic techniques. Namely, (multiple) general $\mathcal{NP}$-reductions, NIZK, non-black-box technique, etc. As a result, there is a gap between plausibility of rZK and solutions for specific useful number theoretic languages. Therefore, it is an important open question to develop new techniques (bypassing the general $\mathcal{NP}$-reductions) that can be implemented with small constant number of (say) exponentiations under widely used hardness assumptions. We explore this central issue in this work as well.

1.1 Our contributions

We achieve the first constant-round concurrently-secure rZK (rZK-CS) arguments in the BPK model (without extra requirement on public keys). More specifically:

[*]The BPK model does allow dynamic key registrations and readers are referred to [6] for the details of dealing with dynamic key registrations.

- For any $\mathcal{NP}$-language, we achieve 5-round rZK-CS arguments for $\mathcal{NP}$ under any preimage-verifiable one-way function (OWF), and 4-round (i.e., *optimal*) rZK-CS arguments for $\mathcal{NP}$ under any one-way permutation OWP and any preimage-verifiable OWF. To this end, we develop the first one-round trapdoor commitment scheme (that is based on any OWP). This construction is of independent interest and can be employed elsewhere (We note that a OWF f is preimage-verifiable if given a string y one can efficiently verifier whether or not there exists a x such that $y = f(x)$. Note that preimage-verifiable OWF is a generic and actually quite weak hardness assumption that includes, in particular, any certified one-way permutation.)

- Beyond plausibility, we present a *generic practical* transformation achieving 5-round rZK-CS arguments in the real BPK model. By "generic" we mean applicability to any language that admits Σ-protocols. By "practical", we mean that our construction does not go through general $\mathcal{NP}$-reductions, and if the starting Σ-protocol and the underlying PRF are practical then the transformed rZK-CS protocol is also practical. (For example, when instantiated with DL or RSA functions, together with the Naor-Reingold practical PRFs, the transformed rZK-CS protocol (for the languages of DL or RSA respectively) employs a very small constant number of exponentiations). This directly implies practical DLP or RSA based identification scheme secure against resetting attack.

A comparisons with the recent work of [13] is presented in Appendix A.

1.2 Organizations

We present preliminary definitions in Section 2. In Section 3, we then present the generic practical protocol (without $\mathcal{NP}$-reductions) transforming Σ-protocols to the rZK protocol in the BPK model. In Section 4, we first present a 5-round rZK-CS arguments for $\mathcal{NP}$ in the BPK model under any preimage-verifiable OWF, and then show how the round-complexity can be further reduced to four (that is optimal) by developing the first one-round OWP-based trapdoor commitment scheme.

2 Preliminaries

In this section, we quickly recall the definitions and the major cryptographic tools used in this work.

Definitions of rZK and concurrent soundness in the BPK model. We assume the reader is familiar with these definitions, and the formal definitions are presented in Appendix C.

Pseudorandom functions PRFs. We also assume the reader is familiar with the definitions of PRF. PRFs can be constructed under any one-way function. The current most practical PRFs are the Naor-Reingold implementations under the factoring (Blum integers) or the decisional Diffie-Hellman hardness assumptions [28]. 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.

Σ-protocols and Σ_{OR}-protocols. The formal presentation of this part is given in Appendix D. The following is informal descriptions.

The idea of Σ-protocols as an abstract concept is introduced by Cramer in [9]. 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 and efficient electronic payment and voting systems. We remark that a very large number of Σ-protocols have been developed in the literature (mainly in applied cryptography), and

Σ-protocol examples for DLP and RSA are given in Appendix D. For a good survey of Σ-protocols and their applications, readers are referred to [12, 10].

One basic construction employing 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 [11]. 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, then we can construct another Σ-protocol, called Σ_{OR}, for the relation $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R_0 \text{ or } (x_1, w) \in R_1\}$. Moreover, for any malicious verifier V^*, the probability distribution of conversations between P and V^*, where w is such that $(x_b, w) \in R_b$, is independent of b. That is, Σ_{OR} is perfectly witness indistinguishable proof of knowledge (the POK property is due to that Σ_{OR}-protocol is itself Σ-protocol).

Perfect-hiding trapdoor commitment TC schemes with additional properties. We recall some perfectly-hiding trapdoor commitment schemes and clarify some additional properties about them which are critical for our purpose (in particular, for the generic but practical transformation from public-coin HVZK to rZK-CS arguments in the BPK model).

The definition of normal (perfectly-hiding) trapdoor commitments is given in Appendix E. As shown in the definition there, known trapdoor commitment schemes work in two rounds as follows: In the first round, the commitment receiver sends the $TCPK$ to the commitment sender. In the second round, on $TCPK$ and the value v to be committed to, the sender computes $(c, d) \leftarrow TCCom(TCPK, v)$ and sends c as the commitment, while keeping the value v and the decommitment information d in private. The trapdoor is the corresponding $TCSK$ with which one can equivocate the commitment at its wish. But, for our purpose, we need TC schemes that satisfy the following additional requirements:

1. Public-key verifiability. The validity of $TCPK$ (even generated by a malicious commitment receiver) can be efficiently verified. In particular, given any $TCPK$, one can efficiently verify whether or not $TCSK$ exists.

2. Public-key Σ-provability. On common input $TCPK$ and private input $TCSK$, one can prove, by Σ-protocols, the knowledge of $TCSK$.

We call a trapdoor commitment scheme satisfying the above two additional properties a *verifiable and Σ-provable trapdoor commitment* (VPTC, in short) scheme. The first round of a VPTC scheme is denoted by $VPTCPK$ and the corresponding trapdoor is denoted by $VPTCSK$. We note both the DLP-based [5] and the RSA-based [29] perfectly-hiding trapdoor commitment schemes are VPTC schemes.

Consider the DLP-based perfectly-hiding trapdoor commitment scheme [5]: On a security parameter n, the receiver selects uniformly an n-bit prime p so that $q = (p-1)/2$ is a prime, an element g of order q in $\mathbf{Z}_p^*$. Then the receiver uniformly selects w in $\mathbf{Z}_q$ and sets $h = g^w \ mod \ p$. The receiver sets $TCPK = (p, q, g, h)$ and keeps w as $TCSK$ in secret. To commit to a value v in $\mathbf{Z}_q$, the sender first checks that: p, q are primes, $p = 2q + 1$ and g, h are elements of order q in $\mathbf{Z}_p^*$ (this in particular guarantees that there exists a w such that $h = g^w \ mod \ p$ as there is a unique subgroup of order q in $\mathbf{Z}_p^*$), otherwise it halts announcing that the receiver is cheating. (This guarantees the public-key verifiability property above.) If the $TCPK$ passes the above testing, then the sender uniformly selects $d \in \mathbf{Z}_q$ (the decommitment information), and sends $c = g^d h^v \mod p$ as its commitment. The public-key Σ-provability is direct from the Σ-protocol for DLP (presented in Appendix D).

Now, consider the RSA-based perfectly-hiding trapdoor commitment scheme [29]: Let N be a composite number and $q > N$ be a prime number, the receiver randomly chooses w from Z_N^* and computes $y = w^q \mod N$. The $TCPK$ is set to be (N, q, y) and $TCSK = w$. To commit to a value $v \in Z_q$, the sender firstly checks that: N is a composite number, $q > N$ is a prime number and y is in Z_N^* (this in particular guarantees the existence of w). If the above checking is successful, then the sender randomly chooses $d \in Z_N^*$ and computes $c = y^v d^q$ as its commitment. The public-key Σ-provability is direct from the Σ-protocol for RSA (presented in Appendix D).

3 The Generic but Practical Transformation from any Σ-Protocol

Let L be any language that admits Σ-protocols (e.g., the language of 2^n residues or the language of a generated subgroup), f_V be any OWF that admits Σ-protocols, and VPTC be a verifiable Σ-provable trapdoor commitment scheme with $VPTCPK$ as its first round and $VPTCSK$ as the corresponding trapdoor. Each (honest) verifier randomly selects x_V from the domain of f_V and publishes $y_V = f_V(x_V)$ as its public-key. On a common input $x \in L$, the following is the high-level overview of the transformation that works in two phases: In the first phase, the prover P first sends $VPTCPK$ to the verifier V, and (if $VPTCPK$ is valid) then V proves to P that it knows either the preimage of y_V or $VPTCSK$, by executing Σ_{OR} on $(VPTCPK, y_V)$. In the second phase, if P accepts the above Σ_{OR}-proof (from V to P), then P proves to V that it knows either the witness of $x \in L$ or the preimage of y_V, by executing Σ_{OR} on (x, y_V) in which the second round (i. e. the random challenge from V to P) is denoted by e_V. Finally, to make the transformed protocol resettable we require that: V first commit to e_V on the top of the above transformation using the underlying VPTC scheme, P and V use different security parameters such that breaking $VPTCPK$ does not compromise the security of y_V, and the randomness of P is generated by applying a PRF on some partial transcript. The details are given in Figure 1 (page 6).

The protocol depicted in Figure-1 runs in 7 rounds, but it can be easily combined into a 5-round protocol in which P sends $VPTCPK$ in the first-round, V sends (c_V, a_V) in the second-round and then the remaining interactions are combined into other three rounds. Note that the transformation does not go through general $\mathcal{NP}$-reductions, and if the underlying VPTC, PRF and the starting Σ-protocol for L are practical, then the transformed protocol is also a practical protocol for L. When instantiated with the DL function for the underlying VPTC and the starting Σ-protocol, together with the Naor-Reingold practical PRF, the transformed protocol (for the language DL) goes through about 12 exponentiations at the rZK prover side and 9 exponentiations at the verifier side.

Theorem 3.1 *Assuming the existence of PRF, VPTC, and OWF that admits Σ-protocol and is secure against subexponentially-strong adversaries, the protocol depicted in Figure-1 is a 5-round rZK-CS argument without $\mathcal{NP}$-reductions in the BPK model for any language that admits Σ-protocols.*

Proof (sketch).

Black-box resettable zero-knowledge.

For any (s,t)-resetting adversary V^* (as defined in Appendix C) who receives $s(n)$ *distinct* strings $x_1, \cdots, x_{s(n)}$ of length n each, and outputs an arbitrary public-file F containing $s(n)$ entries $PK_1, \cdots, PK_{s(n)}$ in its first stage, we first assume the rZK simulator S knows all secret-keys (if such exist) corresponding to public-keys registered in F. Also we assume V^* works in the sequential version (which is equivalent to the interleaving version as discussed in Appendix C) and the real prover uses a truly random function rather than a PRF. S runs V^* as a subroutine and works session by session. In each session j with a distinct "determining" message $(x_k, F, c_V^{(j)}, a_V^{(j)}, PK_i, i)$ $(1 \leq i, j, k \leq s(n))$, then S uses independent random strings in its remaining computation after that (note that the $VPTCPK$ in the first-round can be fixed once and for all), and uses the (assumed known) secret-key SK_i as its witness in the Σ_{OR}-protocol of Phase-2. 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. The indistinguishability between the simulated transcript and the real interaction transcript is from the (perfect) WI property of the Σ_{OR} of Phase-2. Actually, if we remove the Σ_{OR}-protocol of Phase-1, then the remained interactions constitute a constant-round rWI arguments on common input (x, PK_i) (which may be of independent value), as shown by the general paradigm of [6] for achieving rWI from admissible systems. The extension of the proof to the case that the real prover uses a PRF (rather than the assumed truly random

Key Generation. The verifier uses a security parameter N, on which 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.

Common input. The public-file F, an element $x \in L \cap \{0,1\}^n$ and index i that specifies the i-th entry of F, PK_i.

P **private input.** An $\mathcal{NP}$-witness *wit* for $x \in L$, a pair of random strings (w_1, w_2) of length n each, where w_1 is used to generate $VPTCPK$ and w_2 is the randomness seed of a PRF. Note that the security parameter used by P is n.

V **private input.** Private key SK_i such that $PK_i = f_V(SK_i)$.

Complexity-leverage used. Let c_1, $0 < c_1 < 1$, be the constant that the one-wayness of the OWF f (used by V) holds against any circuit of size $2^{N^{c_1}}$. Let c_2 be the constant that: for all sufficiently large n, both the length of the witness *wit* for $x \in L \cap \{0,1\}^n$ and the length of $VPTCSK$ are bounded by n^{c_2}. Then we set $\epsilon > c_2/c_1$ and $N = n^{\epsilon}$. This ensures that one can enumerate all potential witnesses *wit*, or all potential $VPTCSK$s in time $2^{n^{c_2}}$, which is still lesser than the time it would take to break the one-wayness of f (because $2^{n^{c_2}} < 2^{N^{c_1}}$).

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

Stage-1. P generates $VPTCPK$ on security parameter n by using the randomness w_1, and sends $VPTCPK$ to V.

Stage-2. V first checks the validity of $VPTCPK$ received and aborts if it is not valid (this is guaranteed by the public-key verifiability of the underlying VPTC scheme). Otherwise (i.e. $VPTCPK$ is valid), V randomly chooses a random string e_V of length t and computes $c_V = VPTCCom(VPTCPK, e_V)$ (that is, V commits to e_V using the underlying VPTC scheme). V sends c_V to P, and proves to P that it knows either the preimage of PK_i (i.e. SK_i) or $VPTCSK$, by executing the Σ_{OR}-protocol on $(PK_i, VPTCPK)$ 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 of this Σ_{OR}-protocol, then all randomness used by P (from then on after receiving (c_V, a_V)) in the remaining computation is got by applying $PRF(w_2, \cdot)$ on $(x, F, c_V, a_V, PK_i, i)$. If V successfully finishes the Σ_{OR}-protocol and P accepts, then goto Phase-2. Otherwise, P aborts.

Phase-2. P proves to V that it knows either the witness *wit* for $x \in L$ or the preimage of PK_i, by executing the Σ_{OR}-protocol on (x, PK_i) in which V sends the second-round message (i.e. the assumed random challenge from V) by just revealing e_V committed in c_V. We also denote by a_P, z_P the first-round message and the third-round message of this Σ_{OR}-protocol, respectively.

Figure 1. Generic but Practical rZK-CS arguments for any language that admits Σ-protocols

functions) is standard, and the standard technique of [6, 1] can be directly used to extract all secret-keys (if such exist) corresponding to the public-keys in F in expected polynomial-time. Note that although the Σ_{OR} in Stage-2 of Phase-1 is with respect to the common input $(VPTCPK, PK_i)$, but the whole interactions of Phase-1 constitute an *argument of knowledge of the secret-key* SK_i (as $VPTCPK$ is sent by the honest prover). Details of the rZK simulation from scratch 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 special soundness of the Σ_{OR}-protocol of Phase-1 that holds with respect to *any* common input (in particular, *any* public-key registered by V^*, whether valid or not). That is, our protocol works in the real BPK model.

Concurrent soundness.

Suppose the transformed protocol depicted in Figure-1 does not satisfy concurrent soundness in the BPK model, then according to the definition of concurrent soundness in the BPK model (described in Appendix C), there exists an s-concurrent malicious prover P^* such that in a concurrent attack issued by P^* against an honest verifier V with public-key y_V, with non-negligible probability $q(n)$ there exists a j, $1 \leqslant j \leqslant s(n)$, such that V outputs "accept x_j" in session j while $x_j \notin L$. Then we will construct an algorithm E that takes the public-key y_V as input and outputs the corresponding secret-key x_V with non-negligible probability $\frac{(q(n))^2}{s(n)}$ in time $poly(n) \cdot 2^{n^{c_2}} < 2^{N^{c_1}}$, which breaks the hardness assumption on the OWF f_V used by V.

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 y_V. In each session, after receiving a $VPTCPK$ E extracts the corresponding $VPTCSK$ by brute-force searching in time $2^{n^{c_2}}$, and then sends a VPTC commitment to 0^t (rather than to a random string of length t as the honest verifier does), uses $VPTCSK$ as the witness in the Σ_{OR}-protocol of Phase-1 of that session, and decommits the VPTC commitment to a random string of length t by using the trapdoor $VPTCSK$ in Phase-2 of that session. In the j-th session, denote by $c_V^{(j)}$ the VPTC commitment (that commits to 0^t) sent by E in the second round, then in the Phase-2 of the j-th session after receiving $a_{P^*}^{(j)}$ (the first-round message of Phase-2 of the j-th session), E decommits $c_V^{(j)}$ to a random string e_V of length t. Whenever E receives a valid $z_{P^*}^{(j)}$ such that $(a_{P^*}^{(j)}, e_P, z_{P^*}^{(j)})$ is an accepting conversation of the Σ_{OR} of Phase-2 on (x_j, y_V), E rewinds P^* to the point that it just sent $a_{P^*}^{(j)}$, and decommits $c_V^{(j)}$ to a new random string $e'_V \neq e_V$ (by using the trapdoor $VPTCSK$) and runs P^* further. Whenever E receives again a valid $z_{P^*}^{(j)\prime}$ and $(a_{P^*}^{(j)}, e'_V, z_{P^*}^{(j)\prime})$ is an accepting conversation on (x_j, y_V), then E stops.

Due to the public-key verifiability, the perfectly hiding and trapdoorness properties of the underlying VPTC scheme, and the perfect WI property of the Σ_{OR} of Phase-1, with the same probability $q(n)$ there exists an j, $1 \leq j \leq s(n)$, such that P^* can convince E of a false $x_j \notin L$ in the j-th session, *even E sends VPTC commitments to 0^t (rather than to random strings of length t) and uses the extracted $VPTCSK$ (rather than x_V as used by the honest verifier) as its witness in the Σ_{OR} of Phase-1.* Conditioned on E correctly guessed the value j, then with probability $(q(n))^2$ E will extract either the witness wit for $x_j \in L$ or the secret-key x_V such that $y_V = f_V(x_V)$, which is guaranteed by the special soundness of the Σ_{OR} of Phase-2. As we assume $x_j \notin L$ and E randomly guesses j from $\{1, \cdots, s(n)\}$, we conclude E outputs x_V with probability $\frac{(q(n))^2}{s(n)}$. Note that E works in $poly(n) \cdot 2^{n^{c_2}} < 2^{N^{c_1}}$ time, which violates the hardness assumption on the OWF f_V used by V.

Comments: The public-key verifiability property of the underlying VPTC scheme is necessary for the above concurrent soundness proof, as otherwise, as discussed in Appendix B, the simulation of E will be distinguishable from the real interactions between P^* and the honest verifier. We remark that the complexity leveraging technique plays a critical role in the above proof of concurrent soundness. Note that when E rewinds the concurrent malicious P^* in the above proof, E itself is also rewound as P^* is concurrently interacting with it, from which the witness (i.e. $VPTCSK$) used by E in the Σ_{OR} of Phase-1 may be exposed. This is just the place the underlying complexity leveraging plays its role, which says that breaking (or exposing) $VPTCPK$ does not compromise the security of the OWF f_V used by the verifier. Actually, we do not know how to prove concurrent soundness under standard polynomial hardness assumptions without complexity leveraging. $\square$

4 The Theoretical Constructions for $\mathcal{NP}$

In this section, we show how to achieve rZK-CS arguments for any language in $\mathcal{NP}$ in the BPK model, with $\mathcal{NP}$-reductions but with weaker hardness assumptions and reduced (optimal) round-complexity.

4.1 5-Round rZK-CS Arguments for $\mathcal{NP}$ in the BPK Model under any Preimage-Verifiable One-Way Function

First note that preimage-verifiable OWF is a generic and much weaker hardness assumption. In particular, any certified one-way permutation is a preimage-verifiable OWF. Before describing the construction, we first recall some generic tools used.

4.1.1 Generic tools used

Perfectly-binding commitments. One-round perfectly-binding (computationally-hiding) commitments can be based on any one-way permutation OWP [3, 20]. And two-round (public-coin) perfectly-binding commitments can be based on any OWF [27], in which the commitment receiver sends a random string of length $3N$ on a security parameter N in the first-round. *Furthermore, 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.*

Blum's protocol for HC [4]. The parallel repetitions of Blum's basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [4] is just a 3-round public-coin WIPOK for $\mathcal{NP}$ under any one-way permutation (as the first round of it involves one-round perfectly-binding commitments (of a random permutation of G) which are based on any one-way permutations [3, 20]). But it can be easily modified into a 4-round public-coin WIPOK for $\mathcal{NP}$ under any OWF by using Naor's two-round (public-coin) perfectly-binding commitment scheme [27]. The description of Blum's protocol for HC is given in Appendix G.

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.

Feige-Shamir two-round trapdoor commitments [17]. Based on Blum's protocol for HC, Feige and Shamir developed a generic two-round (computationally-hiding and computationally-binding) trapdoor commitment scheme [17], 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 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 description of the Feige-Shamir trapdoor commitment (FSTC) scheme is given in Appendix G.

The 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, and 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 soundness below). Again, if the underlying perfectly-binding commitment scheme is secure against subexponential-time adversaries, then both the hiding property and the trapdoorness property of the FSTC scheme hold also against subexponential-time adversaries.

4.1.2 The protocol

The idea is to replace the Σ_{OR}-protocols used in the protocol depicted in Figure-1 by Blum's 4-round OWF-based public-coin WIPOK for $\mathcal{NP}$, replace the practical Naor-Reingold PRF by a general OWF-based PRF, and replace the underlying VPTC scheme by the Feige-Shamir OWF-based two-round trapdoor commitment. Also, in the key-generation phase, the OWF f_V used by the honest verifier V is not required any longer to be one that admits Σ-protocols and V can use *any* OWF f_V in forming its public-key $y_V = f_V(x_V)$. But, for provable security, we need some cares on the implementation details. Specifically, for the underlying FSTC scheme we require the OWF f used in forming its first-round message be a preimage-verifiable OWF (this is necessary for proving concurrent soundness as discussed in Appendix B). Actually, this is also the only place of the protocol that is not based on the minimal hardness assumption of any OWF. For complexity-leveraging, the prover and the verifier use security parameters n and N respectively that are the same as specified in Figure-1, but besides that the f_V used in the key-generation phase is required to be secure against $2^{N^{c_1}}$-time adversaries, we also require both the WI property of Blum's protocol for HC and the trapdoorness and hiding properties of the Feige-Shamir scheme hold against $2^{N^{c_1}}$-time adversaries.

In more details, in the first-round (i.e. Stage-1 of Phase-1), the prover P sends $(y_P = f_P(x_P), G, R)$ to the verifier V, where f_P is a preimage-verifiable OWF, x_P is a string of length n chosen randomly from the domain of f_P, G is a directed graph that is reduced from y_P by the Cook-Levin $\mathcal{NP}$-reduction, R is a random string of length $3N$ serving as the first-round message of Naor's OWF-based commitment scheme [27]. Then (i.e. in the Stage-2 of Phase-1), V firstly checks the validity of (y_P, G) and aborts if they are not valid. Otherwise (i.e (y_P, G) are valid), V randomly chooses a string e_V from $\{0,1\}^n$, computes $c_V = FSTCCom((y_P, G, R), e_V)$, sends c_V to P and proves to P by WIPOK for $\mathcal{NP}$ that it knows either the preimage of y_P or the preimage of y_V (its public-key). After that (i.e. in Phase-2), P proves to V by WIPOK for $\mathcal{NP}$ that it knows either the witness of the common input or the preimage of y_V, in which V sends the random challenge by just revealing the committed e_V. It's easy to check the above resultant theoretical protocol still runs in 5 rounds (after round combinations accordingly).

Theorem 4.1 *Under any preimage-verifiable OWF (used by the prover) that is secure against standard polynomial-time adversaries and any OWFs (used by the verifier) that are secure against subexponential-time adversaries, the above (theoretical) protocol is a 5-round concurrently-sound rZK argument for $\mathcal{NP}$.*

Proof (outline).

Note that we only require that the verifier uses OWFs that are secure against $2^{N^{c_1}}$-time adversaries, which in turn guarantees that the security of verifier's public-key, the WI property of Blum's protocol for HC and the trapdoorness and hiding properties of FSTC scheme all hold against $2^{N^{c_1}}$-time adversaries. For the preimage-verifiable OWF f_P used by the prover (in forming the first-round message of the underlying FSTC scheme), it can be only secure against standard polynomial-time adversaries, as the one-wayness of the preimage-verifiable OWF is only used to guarantee the computationally-binding of the underlying FSTC scheme against malicious polynomial-time verifiers (in proving black-box resettable zero-knowledge).

Black-box resettable zero-knowledge.

The proof of black-box rZK for the above theoretical protocol is the same as that for the protocol of Figure-1. Again, we remark that the above theoretical protocol works in the real BPK model as we requires nothing about public-keys registered by malicious verifiers.

Concurrent Soundness.

The proof of concurrent soundness is a bit more complicated than that for the protocol of Figure-1. Suppose in the concurrent attack issued by a s-concurrent malicious P^* against an honest verifier V with public-key y_V, with non-negligible probability $q(n)$ there exists a j, $1 \leqslant j \leqslant s(n)$, such that V outputs "accept x_j" in session j while $x_j \notin L$. The knowledge-extractor E (that on common input y_V runs P^* as a subroutine to output the preimage of y_V) works in the same way as described in Theorem 3.1. Now, we want to argue that P^* will also convince E of a false statement in one of the $s(n)$ sessions with probability $p(n)$ that is negligibly close to $q(n)$. This is trivial in Theorem 3.1 due to the perfectly-hiding property of the underlying VPTC scheme and the perfect WI property of Σ_{OR}. But, for the above theoretical protocol case, both the hiding and trapdoorness properties of the underlying FSTC scheme and the WI property of the Blum's WIPOK for $\mathcal{NP}$ are only *computationally* secure (against $2^{N^{c_1}}$-time adversaries). This is overcome by standard hybrid technique with a critical use of the underlying complexity-leveraging.

Specifically, we consider a hybrid experiment, in which an probabilistic polynomial-time (PPT) algorithm $\hat{E}$ takes (y_V, x_V) as input such that $y_V = f_V(x_V)$ (that is, $\hat{E}$ takes both the verifier's public-key and the corresponding secret-key as the input) and works in the same way as the knowledge-extractor E does but with the following modification: $\hat{E}$ uses x_V as its witness in Stage-2 of Phase-1 of any session just as the honest verifier does. Note that the difference between the interactions between P^* and the honest verifier V and the interactions between P^* and $\hat{E}$ is that: in the real interactions between P^* and V, V always commits to (and accordingly decommits to) a random string of length n using the underlying FSTC scheme, but in the interactions between P^* and $\hat{E}$, $\hat{E}$ always commits to 0^n and then decommits to a random string of length n by using the brute-force extracted trapdoor x_{P^*} (as E does). The difference between the interactions between P^* and E and the interactions between P^* and $\hat{E}$ is that: E always uses the brute-force extracted x_{P^*} as its witness in Stage-2 of Phase-1 of each session, but $\hat{E}$ always uses the verifier's secret-key x_V as its witness (just as the honest verifier does).

Denote by $\hat{q}(n)$ the probability that P^* can convince $\hat{E}$ of a false statement in one of the $s(n)$ sessions, then if $|q(n) - \hat{q}(n)|$ is non-negligible, we can break the hiding and trapdoorness properties of the underlying FSTC scheme in the following way: We runs P^* as a subroutine and interact with a player (who is either the honest verifier V or $\hat{E}$), and for each common statement selected by P^* we verify its verity by just working in $2^{n^{c_2}}$ time. Whenever we find P^* successfully convinces a false statement we output 1, otherwise we output 0. Clearly, by standard hybrid technique, if $|q(n) - \hat{q}(n)|$ is non-negligible we can break the hiding and trapdoorness properties of the underlying FSTC scheme in time $poly(n) \cdot 2^{n^{c_2}} < 2^{N^{c_1}}$. Similarly, we can also prove $|\hat{q}(n) - p(n)|$ is negligible, as otherwise we can break the WI property of Blum's protocol for $\mathcal{NP}$ in time $poly(n) \cdot 2^{n^{c_2}} < 2^{N^{c_1}}$. Finally, conditioned on $|q(n) - p(n)|$ is negligible, the rest of the proof is the same as described in Theorem 3.1. □

4.2 rZK-CS Arguments for $\mathcal{NP}$ with *Optimal* Round-Complexity in the (Real) BPK Model

For the 5-round theoretical protocol developed in Section 4.1, if the verifier$V$ uses a OWP-based one-round perfectly-binding commitment scheme then the prover only needs to send (y_P, G) in the first-round. To further reduce the round-complexity, we want to combine (y_P, G) into the third-round (that is from the prover to the verifier), thereby obtaining 4-round (that is optimal) rZK-CS arguments for $\mathcal{NP}$. Recall that (y_P, G) is used by V in two ways: On one hand, it forms the $\mathcal{NP}$-statement (to be precise, a directed graph reduced from (y_P, y_V) by $\mathcal{NP}$-reduction) to be proved by V by Blum's WIPOK for HC in Stage-2 of Phase-1; on the other hand, it serves $TCPK$ of the underlying FSTC scheme with x_P (or equivalently, a Hamiltonian cycle of G) as the trapdoor $TCSK$. To combine (y_P, G) 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 HC graph G other than the lower and upper bounds of its size (guaranteed by the underlying $\mathcal{NP}$-reduction from y_P to G), and G is only sent in the decommitment stage after the commitment stage is finished.

For the first cryptographic tool of above, we note that the Lapidot-Shamir OWP-based 3-round WIPOK for HC [24] (also described in [19]) is just the protocol of the type we need. 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. But, it can be easily extended to the case that the prover knows only the lower and upper bounds of the size of the graph to be proved. Details are given in Appendix H.

Thus, the big 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 to a bit 0, the committer sends an n-by-n adjacency matrix of commitments with each entry of the adjacency matrix committing to 0. To commit to a bit 1, the committer sends an n-by-n 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 $|V| = n$ 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 at the $(\pi(i), \pi(j))$ position 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 cycle of n).

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 to both 0 and 1 for the same commitment-stage message 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 $p(n)$ and the upper-bound $q(n)$ of the size of G. In this case, in commitment stage P sends $(q(n) - p(n) + 1)$ many adjacency matrices with vertex-sizes ranging from $p(n)$ to $q(n)$. In the decommitment stage, after the size of G is clear, P only decommits with respect to the unique adjacency matrix of according size.

Comments: Although the above one-round trapdoor commitment scheme is developed here to reduce round-complexity for rZK, but we remark that it is of independent value and, in particular, can be used to reduce round-complexity of other cryptographic protocols involving trapdoor commitments.

Finally, using almost the same proof procedure of Theorem 4.1, we can prove the following theorem:

Theorem 4.2 *Under 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 subexponential-time adversaries, any language in $\mathcal{NP}$ has a 4-round (that is optimal) rZK-CS argument. In particular, this implies 4-round rZK-CS arguments for $\mathcal{NP}$ can be implemented with any certified permutation.*

References

[1] 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

[2] 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.

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

[4] 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.

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

[6] 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/

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

[8] 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.

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

[10] 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

[11] 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.

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

[29] 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.

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

[31] 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.

Appendix A. Comparisons with the work of [13]

The rZK-CS protocol of [13] employs a stronger version of the BPK model (specifically, the certified public-key model); and in itself, achieving the result in the stronger model is justified and interesting due to the failure so far to achieve it in the waker model. In the protocol of [13], the verifier's public-key (registered in the public-file) serves as the public-key of the underlying public-key encryption scheme used by the verifier. But, the proof of rZK in [13] critically depends on the following additional requirements on public-keys generated by even malicious verifiers.

1. The validity of public-keys generated by malicious verifiers can be efficiently verifier.
2. Any ciphertext generated with a valid public-key corresponds to at most one plaintext.

The underlying PKE is actually used by the verifier as a perfectly-binding commitment scheme in a coin-tossing subprotocol with its output, in turn, serving as the common random string for the underlying NIZK from the prover to the verifier. If the above additional requirements on public-keys generated by malicious verifiers do not hold (and the scheme is used in the bare PUBLIC KEY model), then a malicious verifier will have the potential ability to set the output of the coin-tossing protocol by which it can extract the prover's private $\mathcal{NP}$-witness in case that the underlying NIZK is NIZKPOK.

The second point of difference between the works is that the hardness assumptions used in [13] are not generic and are stronger than those used in our protocols for NP. The protocol of [13] assumes any *certified* OWP and a special form of PKE with known implementation based on DLP. (The paper does not explicitly assumes more than "any OWP," but one can observe that a certified OWP is necessary for provable security there. This is detailed next in Appendix B.

The third point of difference, is that the earlier work only considered general plausibility for $\mathcal{NP}$ languages (going through multiple $\mathcal{NP}$-reductions and using general NIZK).

Appendix B. A Note on the Underlying Hardness Assumptions of [13]

The protocol of [13] is based on the existences of any one-way permutation OWP and a special form of public-key encryption scheme with known implementation based on DLP. The OWP hardness assumption is due to the underlying NIZK and the "puzzle" sent from the prover to the verifier. Specifically, the puzzle is $y = f(x)$ for a OWP f. But, assuming that f is a OWP is not enough for provable security.

In proving concurrent soundness of the protocol in [13], for the "puzzle" y of length n sent from a malicious prover P^*, the knowledge extractor E (that runs P^* as its subroutine by playing the role of the honest verifier V) first extracts the preimage x of y under f by brute-force searching in 2^n time, and then uses the extracted x (if such exists) as its witness to finish its simulation. For proving concurrent soundness, it is required that the simulation of E is indistinguishable from the interactions between P^* and the real honest verifier V.

The subtle observation here is that by assuming only that f is a OWF, it is not guaranteed that given any y there exists a x such that $f(x) = y$. For example, consider the SQUARE one-way permutation (over the quadratic residues): $f(x) = x^2 \mod N$, where $N = p \cdot q$ and $p = q = 3 \mod 4$. If the underlying OWP used in [13] is the above SQUARE OWP, then the malicious prover P^* may form the puzzle y which is a non-square such that there exists no x satisfying $y = x^2 \mod N$. Note that the real honest verifier (which is a PPT algorithm) cannot verify whether y is a square or not, and thus always finishes its interactions with P^*. But, in the simulation of E, E cannot extracts the preimage of y and thus cannot finish the simulation, which is clearly distinguishable from the *finished* interactions between P^* and the real honest verifier. From the above clarifications, for provable security of [13] we need to require that for any puzzle y sent by P^* it is easy to verify whether or not the corresponding preimage exists. This can be achieved by additionally requiring that the OWP used by the prover in forming the puzzle to be a "*certified*" one. A permutation family is certified if it is easy to verify (in polynomial time) that a given function belongs to the family.

Appendix C. Definitions of rZK and Concurrent Soundness in the BPK 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.
- $P(1^n, x, y, F, id, w)$ be an honest prover that is a *polynomial-time* interactive machine, where 1^n is a security parameter, x is an n-bit string in L, y is an auxiliary input, F is a public-file, id is a verifier identity, and w is his random-tape.
- V be an honest verifier that is an 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 public-key PK and remembers the corresponding secret key SK.
    2. Verification stage. V, on inputs SK, $x \in \{0,1\}^n$ and a random tape ρ, performs an interactive protocol with a prover and outputs "accept x" or "reject x".

The malicious resetting verifier and resettable zero-knowledge

A malicious (s,t)-resetting malicious verifier V^*, where t and s are positive polynomials, is a $t(n)$-time TM working in two stages so that, on input 1^n,

Stage 1 V^* receives $s(n)$ *distinct* strings $x_1, \cdots, x_{s(n)}$ of length 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, $w_1, \cdots, w_{s(n)}$, are randomly selected and then fixed for P, resulting in $s(n)^3$ deterministic prover strategies $P(x_i, id_j, w_k)$, $1 \leq i, j, k \leq s(n)$. V^* is then given oracle access to these $s(n)^3$ provers, and finally outputs its "view" of the interactions (i. e. its random tape and messages received from all his oracles).

Definition C.1 (Black-box resettable zero-knowledge) *A protocol $< P, V >$ is black-box resettable zero-knowledge for a language $L \in \mathcal{NP}$ if there exists a black-box simulator M such that for every (s,t)-resetting verifier V^*, the following two probability distributions are indistinguishable. Let each distributions be indexed by a sequence of common* **distinct** *inputs $\bar{x} = x_1, \cdots, x_{s(n)} \in L \cap \{0,1\}^n$ and their corresponding NP-witnesses $aux(\bar{x}) = y_1, \cdots, y_{s(n)}$:*

Distribution 1 *The output of V^* obtained from the experiment of choosing $w_1, \cdots, w_{s(n)}$ 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: $P(x_i, y_i, F, id_j, w_k)$ 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 $M^{V^*}(x_1, \cdots, x_{s(n)})$.*

Remark. In Distribution 1 above, since V^* oracle accesses to $s(n)^3$ instances P: $P(x_i, y_i, F, id_j, w_k)$, $1 \leq i, j, k \leq s(n)$, it means that V^* may invoke and interact with the same $P(x_i, y_i, F, id_j, w_k)$ multiple times, where each such interaction is called a session. We remark that there are two versions for V^* works 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 interaction with the current oracle $P(x_i, y_i, F, id_j, w_k)$ before starting an interaction with any $P(x_{i'}, y_{i'}, F, id_{j'}, w_{k'})$, 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 his oracles and conducts a complete interaction with it.

2. Interleaving version. In this version the above restriction is removed and so V^* may initiate and interact with $P(x_i, y_i, F, id_j, w_k)$s concurrently in many sessions. That is, we allow V^* to send arbitrary messages to each of the $P(x_i, y_i, F, id_j, w_k)$ and obtain the response of $P(x_i, y_i, F, id_j, w_k)$ to such message.

However, these two versions are equivalent as shown in [6]. In other words, interleaving interactions do not help the malicious verifier get more advantages on learning knowledge from his oracles than he 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.

Concurrent soundness in the BPK model

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

If P^* is already running $i-1$ $(1 \leq i-1 \leq s(n))$ sessions, it can select *on the fly* a common input $x_i \in \{0,1\}^n$ (which may be equal to x_j for $1 \leq j < i$) and initiate a new session with the verification stage of $V(SK, x_i, \rho_i)$. We stress that in different sessions V uses independent random-tapes in his verification stage (that is, $\rho_1, \cdots, \rho_i$ $(1 \leq i \leq s(n))$ are independent random strings).

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

Appendix D. Σ-Protocols and Σ_{OR}-Protocols

Σ-protocol

A 3-round public-coin protocol $< P, V >$ 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 computes 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 same probability distribution as the real conversation between the honest P, V on input x.

Σ-protocol for DLP:
The following is a Σ-protocol $< P, V >$ proposed by Schnorr [30] 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 prime and that g, h have order q, and accepts iff this is the case.

Σ-Protocol for RSA [22]:
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 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 [11].

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 s and sends it to P.
- P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b to challenge e_b using (x_b, a_b, e_b, w) as input. He sends (e_0, z_0, e_1, z_1) to V.
- V checks that $s = e_0 \oplus e_1$ and that conversations (a_0, e_0, z_o), (a_1, e_1, z_1) are accepting conversations with respect to inputs x_0, x_1, respectively.

Theorem D.1 [12] *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, for any malicious verifier V^*, the probability distribution of conversations between P and V^*, where w is such that $(x_b, w) \in R_b$, is independent of b. That is, Σ_{OR} is perfectly witness indistinguishable.*

Appendix E. The definition of perfectly-hiding trapdoor commitment TC scheme

Definition E.1 (perfectly-hiding trapdoor commitment scheme TC) *A trapdoor commitment scheme (TC) is a quintuple of probabilistic polynomial-time (PPT) algorithms TCGen, TCCom, TCVer, TCKeyVer and TCFake, such that*

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

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

Appendix F. The rZK Simulation from Scratch

Note that in the proof of Theorem 3.1, we have assumed that the rZK simulator knows all secret-keys (if such exist) corresponding to public-keys registered by the (s, t)-resetting malicious verifier V^* in the public-key file F. But, the reality is that the rZK simulator S does not know such secret-keys from the beginning of the simulation. This problem is overcome by the standard technique of [6].

First of all, without loss of generality, we make 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, we assume the real prover uses a truly random function (rather than a PRF), as the proof can be easily extended, by standard hybrid technique, to the case when the real prover uses a PRF.

For any (s, t)-resetting adversary V^* (as defined in Appendix C) who receives $s(n)$ *distinct* strings $x_1, \cdots, x_{s(n)}$ of length n each, 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_i in F, $1 \leq i \leq s(n)$, is "covered" if the rZK simulator S has already learnt the corresponding secret-key SK_i (if such exists).

The rZK simulator S works in $s(n)+1$ phases such that in each phase it either successfully finishes its simulation or "covers" a new public-key in F. In each phase, S runs V^* as a subroutine and works session by session sequentially. For any session with respect to a "covered" public-key, then there is no problem for S to successfully finish the simulation of this session (as described in the proof of Theorem 3.1). The difficulty occurs in simulating the Phase-2 of a session j with "determining" message $(x_k, F, c_V^{(j)}, a_V^{(j)}, PK_i, i)$ $(1 \leq i, j, k \leq s(n))$ with respect to an uncovered public-key PK_i, as S has not yet learned the secret-key SK_i and also S has no the $\mathcal{NP}$-witness of the common input x_k (possessed by the real prover). The approach for S is to extract the corresponding secret-key SK_i in polynomial time (if such exists) and then go to the next phase where it simulates the simulation from scratch again but with one more covered public-key. Recall that although the Σ_{OR} in Stage-2 of Phase-1 is with respect to the common input $(VPTCPK, PK_i)$, but the whole interactions of Phase-1 constitute an *argument of knowledge of the secret-key* SK_i (as $VPTCPK$ is sent by the honest prover). By the proof of knowledge property (i.e. special soundness) of Σ_{OR}, to extract the corresponding SK_i, S needs to send a new random challenge (i.e. the second-round message of the Σ_{OR} of Phase-1). But the problem here is that such random challenge may have been fixed (determined) by a (possibly) past partial transcript (in which the determining message $(x_k, F, c_V^{(j)}, a_V^{(j)}, PK_i, i)$ appeared at the first time) due to the underlying random function used. This problem is bypassed precisely as in [6, 1]. That is, S rewinds V^* to the point that V^* just sent $(x_k, F, c_V^{(j)}, a_V^{(j)}, PK_i, i)$ *at the first time*, and sends back a new random challenge by using a new random function.

Finally, the same analysis in [6, 1] shows that the above simulation strategy ends in expected polynomial-time and returns a distribution indistinguishable from the real interactions between V^* and honest prover instances.

Appendix G. Blum's Protocol for HC and the Feige-Shamir Trapdoor Commitment Scheme

Blum's protocol for HC.

In the main text, we use the parallel repetitions of the following basic proof system for the directed Hamiltonian Cycle (HC) problem which is $\mathcal{NP}$-Complete.

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

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

Round-1. The prover selects a random permutation, π, of the vertices V, and commits (using a perfectly-binding commitment scheme) to 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.

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 $(i, j) \in C$ (and the verifier checks that all revealed values are 1 and the corresponding entries form a simple q-cycle).

The Feige-Shamir trapdoor commitment scheme.

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 HC, a graph G (with q nodes), such that finding a Hamiltonian cycle in G is equivalent to finding the preimage of x of y. Finally, it sends (y, G) to the commitment sender. We remark that to get OWF-based trapdoor commitments, the commitment receiver also sends a random string R of length $3N$.

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 to 0, the committer commits to a random permutation of G using the underlying perfectly-binding commitment scheme Com (and decommits by revealing the entire graph and the permutation); To commit to 1, the committer commits to a graph containing a randomly labeled q-cycle only (and decommits by opening this cycle only).

The trapdoorness property of the above Feige-Shamir commitment scheme is from the following observation: After sending a commitment to 0, one can decommit to 0 in the same way above. However, it is also possible to decommit it to 1 if one knows a Hamiltonian cycle in G.

Appendix H. The Lapidot-Shamir WIPOK for HC.

The Lapidot-Shamir WIPOK for HC is the n-repetition of the following basic protocol (which is almost verbatim from [23]).

Round-1. The prover P commits to a adjacency matrix for a randomly-labelled cycle C of size n (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 of size n 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 cycle). If $b = 1$, then P sends a random permutation π mapping C_G (i.e. its private witness) to C (committed in its first-round message), and for each non-edge in G, P opens the committed value (that should be 0) at the corresponding position (via the permutation π) of the adjacency matrix committed in the first-round message (and V checks all revealed values are 0 and the unrevealed positions in the committed adjacency matrix constitute a graph that is isomorphic to G via the permutation π).

In the above description, we have assumed P knows the size of the graph G to be proved. But, it can be easily extended to the case that P only knows the lower-bound $p(n)$ and the upper-bound $q(n)$ of the size of G. In this case, in the first-round P commits to $(q(n) - p(n) + 1)$ many cycles with sizes ranging from $p(n)$ to $q(n)$. In the third-round, after the size of G is clear, P only decommits with respect to the unique cycle of according size.