J Cryptol (2016) 29:156–219
DOI: 10.1007/s00145-014-9191-z

Journal of CRYPTOLOGY

Concurrent Knowledge Extraction in Public-Key Models[*]

Andrew Chi-Chih Yao
Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China
andrewcyao@tsinghua.eud.cn

Moti Yung
Google Inc., Mountain View, CA, USA
Columbia University, New York, NY, USA
moti@cs.columbia.edu

Yunlei Zhao[†]
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
ylzhao@fudan.edu.cn

Communicated by Goldreich

Received 28 September 2011
Online publication 11 November 2014

Abstract. Knowledge extraction is a fundamental notion, modeling machine possession of values (witnesses) in a computational complexity sense and enabling one to argue about the internal state of a party in a protocol without probing its internal secret state. However, when transactions are concurrent, say over the Internet, with players possessing public keys (as is common in cryptography), assuring that entities "know" what they claim to know, where adversaries may be well coordinated across different transactions, turns out to be much more subtle and in need of re-examination. In such settings, mixing the public-key structure as part of the language and statements is a natural adversarial strategy. Here, we investigate how to formally treat knowledge possession by parties interacting concurrently in the public-key model. More technically, we look into the relative power of the notion of "concurrent knowledge extraction" (CKE) for concurrent zero knowledge (CZK) in the bare public-key (BPK) model, where the language and statements being proved can be dynamically and adaptively chosen by the prover and may be possibly based on verifiers' public keys. By concrete attacks against some existing natural protocols, we first show that concurrent soundness and normal arguments of knowledge do not guarantee concurrent verifier security in the public-key setting. Here, roughly speaking, concurrent verifier security says that the malicious concurrent prover should "know" all the witnesses to all the *possibly public-key-related* statements adap-

[*] This research was supported in part by the National Basic Research Program of China (973 Program) Nos. 2007CB807900, 2007CB807901, 2014CB340600; National Natural Science Foundation of China Grant Nos. 61472084, 61272012, 61033001, 61061130540, 61361136003; Innovation Project (No. 12ZZ013) of Shanghai Municipal Education Commission; and Joint Project of SKLOLS.

[†] Corresponding author

tively chosen and successfully proved in the concurrent sessions. These concrete attacks serve as a good motivation for understanding "possession of knowledge" for concurrent transactions with registered public keys, i.e., the subtleties of concurrent knowledge extraction in the public-key model. This motivates us to introduce and formalize the notion of CKE, along with clarifications of various subtleties. Two implementations are then presented for constant-round concurrently knowledge extractable concurrent zero-knowledge (CZK–CKE) argument for $\mathcal{NP}$ in the BPK model: One protocol is generic and based on standard polynomial-time assumptions, whereas the other protocol is computationally efficient and employs complexity leveraging in a novel way. Both protocols can be practically instantiated for some specific number-theoretic languages without going through general $\mathcal{NP}$-reductions. Of independent interest are the discussions about the subtleties surrounding the fundamental structure of Feige–Shamir zero knowledge in the BPK model.

Keywords. Proof of knowledge, Zero knowledge, Bare public key, Complexity leveraging, Strong witness indistinguishability, Witness-extended emulator.

1. Introduction

Zero-knowledge (ZK) protocols allow a prover to assure a verifier of validity of theorems without giving away any additional knowledge (i.e., computational advantage) beyond validity. This notion was introduced in [48], and its generality was demonstrated in [44, 45,47]. In particular, the fact that any statement in $\mathcal{NP}$ can be proved in zero knowledge [47] has made ZK protocols widely applicable and particularly playing a central role in protocol design [46,78] (see more in [38,39]). Traditional notion of ZK considers the security in a stand-alone (or sequential) execution of the protocol. Motivated by the use of such protocols in an asynchronous network like the Internet, where many protocols run simultaneously, studying security properties of ZK protocols in such concurrent settings has attracted many research efforts in recent years [2,21,26,32,33]. Informally, a ZK protocol is called concurrent zero knowledge if concurrent instances are all (expected) polynomial-time simulatable, namely when a possibly malicious verifier concurrently interacts with a polynomial number of honest prover instances and schedules message exchanges as it wishes.

The concept of "proof of knowledge" (POK), informally discussed in [48], was then formally treated in [8,9,38,40,41]. POK systems, especially zero-knowledge POK (ZKPOK) systems, play a fundamental role in the design of cryptographic schemes, enabling a formal complexity theoretic treatment of what it means for a machine to "know" something. Roughly speaking, a "proof of knowledge" means that a possibly malicious prover can convince the verifier of an $\mathcal{NP}$ statement if and only if it, in fact, "knows" (i.e., possesses) a witness to the statement (rather than merely conveying the fact that a corresponding witness exists). With the advancement of cryptographic models where parties first publish public keys (e.g., for improving round complexity [16]) and then may choose the statements to prove, knowledge extraction becomes more subtle (due to possible dependency on published keys) and needs re-examination. Here, we investigate the relative power of the notion of "concurrent knowledge extraction" in the concurrent zero-knowledge bare public-key model with adaptive input (language and statements) selection by malicious provers.

The BPK model, introduced in [15], is a natural cryptographic model. A protocol in this model simply assumes that all verifiers have each deposited a public key in a public file (which are referred to as the *key-generation stage*), before user interactions take place (which are referred to as the *proof stage*). No assumption is made on whether the public keys deposited are unique or valid (i.e., public keys can even be "nonsensical," where no corresponding secret keys exist or are known). In many cryptographic settings, availability of a public-key infrastructure (PKI) is assumed or required, and in these settings, the BPK model is, both, natural and attractive (note that the BPK model is, in fact, a weaker version of PKI where in the latter added key certification is assumed). It was pointed out in [64] that the BPK model is, in fact, applicable to interactive systems in general.

Protocol soundness in the BPK model (against malicious provers) turned out to be much more involved than anticipated, as was demonstrated by Micali and Reyzin [64] who 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. Here, we focus on concurrent soundness, which, roughly speaking, means that a possibly malicious probabilistic polynomial-time (PPT) prover P^* cannot convince the honest verifier V of a *false* statement even when P^* is allowed multiple interleaving interactions with V in the public-key model. They also showed that any *black-box* ZK protocol with concurrent soundness in the BPK model (for non-trivial languages outside $\mathcal{BPP}$) must run at least four rounds [64].

Concurrent soundness only guarantees that concurrent interactions cannot help a malicious prover to validate a *false* statement in the public-key model. However, it does *not* prevent a concurrent malicious prover P^* from validating a *true* statement *but* without knowing any witness to the statement being proved, particularly if the statement chosen by P^* is dependent upon verifiers' public keys. Note that, mixing the public-key structure as part of the language and statements can be a natural adversarial strategy against protocols run concurrently in the public-key model. Also, the dependency between the language/statements being proved and verifiers' public keys can be inevitable if the protocols are designed for a set of languages (e.g., all $\mathcal{NP}$ languages via $\mathcal{NP}$-reductions to an $\mathcal{NP}$-complete language, or all languages admitting Σ protocols without going through $\mathcal{NP}$-reductions). This potential vulnerability is not merely a theoretic concern: In fact, most concurrent ZK protocols in the BPK model involve a sub-protocol in which the verifier proves to the prover the knowledge of the secret key corresponding to its registered public key. A malicious prover, in turn, can (as we shall show) exploit these sub-proofs by the verifier in other sessions, without possessing a witness to these sessions' statements that are chosen by the malicious prover possibly based on verifiers' public keys. This issue, in turn, motivates the need for careful definitions and for achieving concurrent verifier security for concurrent ZK in the BPK model for adaptively chosen proofs, so that one can remedy the above security vulnerability. Recall that knowledge extraction, say POK, is fundamental to cryptographic protocols and is required by many cryptographic applications.

Key terminology. *Throughout this paper, whenever we talk of "(concurrent) verifier security" in the public-key model, we refer to the security w.r.t. the "(concurrent) POK property," i.e., the "possession of knowledge" feature against malicious prover who can adaptively set the language and statements to be proved (possibly based on verifiers'* *public keys) and concurrently interact with honest verifiers in the public-key model. That is, the (possibly malicious) concurrent prover should "know" all the witnesses to all the statements adaptively chosen and successfully proved in the concurrent sessions, in the sense that such witnesses can be efficiently extracted (from the internal state of the malicious concurrent prover) and that the extracted witnesses should be "independent" of honest verifiers' secret keys. This notion will be named "concurrent knowledge extraction" in the public-key model.*

1.1. *Our Contributions*

We first investigate the subtleties of concurrent verifier security in the public-key model in the case of proof of knowledge. Specifically, we show concurrent interleaving and malleating attacks against some existing natural protocols running concurrently in the BPK model, which shows that concurrent soundness and normal arguments of knowledge (and also traditional concurrent non-malleability) do not guarantee concurrent verifier security in the BPK model. With these attacks, a malicious prover plays the role of concurrent man-in-the-middle (CMIM) and manages to malleate the interactions with the verifier who proves the knowledge of its secret key in one session into successful interactions with the verifier in another concurrent session *without knowing any witness to the statement being proved*. The separation between traditional POK and CKE in the BPK model is demonstrated w.r.t. any $\mathcal{NP}$-language, while that between concurrent soundness and CKE is demonstrated with some languages dependent upon verifiers' public keys. We emphasize that concurrent soundness holds even when the statements are maliciously chosen by the prover based on the verifiers' public keys. These concrete attacks serve as a good motivation for understanding "possession of knowledge on the Internet with registered public keys," i.e., the subtleties of concurrent knowledge extraction in the public-key model.

Then, we formulate concurrent verifier security that remedies the vulnerability as demonstrated by the concrete attacks which are of the concurrent man-in-the-middle nature, along with detailed subtlety clarifications and discussions. The security notion defined is named `concurrent knowledge extraction (CKE)` in the public-key model, which essentially means that for adaptively chosen statements whose validations are successfully conveyed by a possibly malicious prover to an honest verifier by concurrent interactions, the prover must "know" the corresponding witnesses in a sense that the knowledge "known" by the prover can be efficiently extracted and is "independent" of honest verifier's secret key.[1] We justify our CKE formulation in the public-key model and clarify in detail the various subtleties surrounding the CKE definition, which might be of independent interest. In particular, we show that CKE is strictly stronger than concurrent soundness in the public-key model (assuming the existence of any one-way function).

We then present both generic (based on standard polynomial-time assumptions and employing strong witness indistinguishability [39]) and computationally efficient

[1] We use the name of "concurrent knowledge extraction" to make this notion independent of proof and argument systems. That is, the concept of CKE applies to both proof systems and argument systems, which uniforms and simplifies the notations.

(employing complexity leveraging[2] [15] in a novel way) black-box implementations of constant-round CZK–CKE arguments for $\mathcal{NP}$ in the BPK model. We also show that both the generic and efficient CZK–CKE protocols can be practically instantiated for some number-theoretic languages without going through general $\mathcal{NP}$-reductions. To our knowledge, they are the first provably secure CZK–CKE protocols in the BPK model. Along the way, we clarify in depth the various subtleties surrounding the protocol construction and security analysis of CZK–CKE in the BPK model, which might also be of independent value.

1.1.1. *Overview of CKE Formulation in the Public-Key Model*

The security notion assuring that a malicious prover P^* does "know" what it claims to know, when it is concurrently interacting with the honest verifier V and can set the language and statements to be proved based on verifiers' public keys, can informally be formulated as: for any x, if P^* can convince V (with public key PK) of "$x \in L$" (for an $\mathcal{NP}$-language L that may be chosen dependent on verifiers' public keys) by concurrent interactions, then there exists a PPT knowledge extractor that outputs a witness for $x \in L$. This is a natural extension of the normal arguments of knowledge into the concurrent public-key setting. However, this formulation approach is problematic in the concurrent public-key setting. The reason is the statements being proved may be related to PK (which can be inevitable if L is an $\mathcal{NP}$-complete language), and thus, the extracted witness may be related to the corresponding secret key SK (even just the secret key as shown by our concrete attack on existing natural protocols). However, for *black-box polynomial-time* knowledge extraction (as is the focus of this work), extracting knowledge of the witness from P^* requires a secret SK that P^* does not know. We clarify that building *black-box polynomial-time* knowledge extractors without SK seems impossible (cf. Sect. 5.1). This poses the subtle question: in what sense does P^* know the witness if producing it requires something P^* does not know?

To solve this subtlety, we require the extracted witness, together with adversary's view, to be *independent* of SK. However, the problem here is how to formalize such independence, in particular, w.r.t. a concurrent man-in-the-middle (CMIM) adversary? We solve this in the spirit of non-malleability formulation [31]. That is, we consider the message space (distribution) of SK, and such independence is roughly defined as follows: Let SK be the secret key and SK' be an element randomly and independently distributed over the space of SK, then we require that, for any polynomial-time computable relation R, the probability $\Pr[R(\bar{w}, SK, view) = 1]$ be negligibly close to $\Pr[R(\bar{w}, SK', view) = 1]$, where $\bar{w}$ is the set of witnesses extracted by the knowledge extractor for successful concurrent sessions and $view$ is the view of P^*. This captures the intuition that P^* does, in

[2] Roughly speaking, by complexity leveraging, we specify two different parameters n and n' for two cryptographic primitives f and f', respectively, such that n and n' are polynomially related (i.e., any quantity that is a polynomial of n is also another polynomial of n', and vice versa) but $2^{n'} \ll 2^{n^c}$ for some constant $c, 0 < c < 1$. If we assume f is secure against 2^{n^c}-time adversaries (i.e., sub-exponentially secure), then we can break the security of f' in brute-force in time $2^{n'}$, while still not violating the security of f. This technique of "complexity leveraging" was first introduced in [15] (and then used in a list of subsequent works) for achieving resettable ZK in the BPK model. In this work, complexity leveraging is employed as a paradigm to frustrate CMIM attacks.

fact, "know" the witnesses to the statements whose validations are successfully conveyed by concurrent interactions. Motivation and formal definition of CKE in the public-key setting, along with in-depth detailed clarifications and justifications, are presented in Sects. 4 and 5.

1.1.2. *Overview of Achieving CZK–CKE in the BPK Model*

The starting point is the basic and central Feige–Shamir ZK (FSZK) structure [36]. The FSZK structure is conceptually simple and is composed of two witness-indistinguishable proof of knowledge (WIPOK) sub-protocols. In more detail, letting f be a OWF, in the first WIPOK sub-protocol with the verifier V serving as the knowledge prover, V computes $(y_0 = f(s_0), y_1 = f(s_1))$ for randomly chosen s_0 and s_1; then, V proves to the prover P the knowledge of the preimage of either y_0 or y_1. In the second WIPOK sub-protocol with P serving as the knowledge prover for an $\mathcal{NP}$-language L, on common input x, P proves to V the knowledge of either a valid $\mathcal{NP}$-witness w for $x \in L$ or the preimage of either y_0 or y_1. FSZK is also argument of knowledge and can be instantiated practically (without going through general $\mathcal{NP}$-reductions) by the Σ_{OR} technique [20,82].

Letting (y_0, y_1) serve as the public key of V and s_b (for a random bit b) as the secret key, the public-key version of FSZK is CZK in the BPK model [82]. However, we show that the public-key version of FSZK is not of concurrent soundness [81], let alone of concurrent knowledge extractability.[3] We hope to add the CKE property to FSZK in the BPK model (and thus get concurrent security both for the prover and for the verifier simultaneously), while maintaining its conceptual simplicity and computational efficiency.

The subtle point is: We are actually dealing with a CMIM attacker who manages to malleate, in a malicious and unpredictable way, the public keys and knowledge proof interactions of the verifier in one session into the statements and knowledge proof interactions in another concurrent session. To add CKE security to FSZK in the BPK model, some non-malleable tools appear to be required. Here, we show how to do so without employing such tools.

The crucial idea behind achieving our goal is to strengthen the first sub-protocol to be *statistical* WIPOK, and require the prover to first, before starting the second WIPOK sub-protocol, commit to the supposed witness to c_w by running a *statistically binding* commitment scheme. This guarantees that if the witness committed to c_w is dependent on the secret key used by V, there are, in fact, certain differences between the interaction distribution when V uses $SK = s_0$ and the one when V uses $SK = s_1$. We can, in turn, use such distribution differences to violate the statistical WI of the first sub-protocol, which then implies *statistical* CKE. This solution, however, loses CZK in general, since the second WI sub-protocol is run w.r.t. commitments to different values in real interactions and in the simulation. Specifically, the composition of *statistically binding* commitment and regular WI does not preserve the regular WI property. To deal with this problem, we

[3] Indeed, FSZK was not designed for the public-key model. But FSZK in the BPK model serves as the basis for a list of constant-round concurrent and resettable ZK works in the BPK model [17,24,25,28,68,75,77,79–81]. We suggest it may be useful to make this issue clear (for better understanding and for correct deployment of FSZK in the public-key model).

employ a stronger second sub-protocol, i.e., strong WI argument/proof of knowledge (strong WIPOK) [38].[4] We show that composing statistically binding commitment and SWI yields a regular WI, and thus, the CZK property is retained.

Employing SWI complicates the protocol structure and incurs protocol inefficiency. It is, therefore, desirable to keep using *any* regular WIPOK in the second sub-protocol, for conceptual simplicity and efficiency. To bypass the subtleties of SWI for the CZK proof, we employ a double-commitment technique. Specifically, we require the prover to produce a pair of statistically binding commitments, c_w and c_{sk}, before starting the second WIPOK sub-protocol of FSZK, where c_w is supposed to commit to a valid $\mathcal{NP}$-witness for $x \in L$ and c_{sk} is supposed to commit to the preimage of either y_0 or y_1. In real interactions, c_{sk} actually commits to 0 and the honest prover uses what is committed to c_w as its witness. However, in the CZK simulation, the simulator tries to first extract the verifier's secret key and then commit it to c_{sk}. Double commitments can bypass, by hybrid arguments, the subtleties of SWI for the CZK proof. However, the provable CKE property with double commitments turns out to be much subtler. Specifically, due to the double commitments used, the value extracted can be either the value committed to c_w or that to c_{sk}. If it is ensured that the valued extracted is always the one committed to c_w, we can get statistical CKE in the same way as the SWI-based solution. By the one-wayness of f, the value extracted in polynomial time cannot be the preimage of y_{1-b} (recall the secret key is s_b). However, how about the possibility that the value extracted is just the secret key s_b committed to c_{sk}? Consider the following adversarial strategy:

Imagine that the malicious prover P^* commits (w, s_0) to (c_w, c_{sk}) in some session, where w is a valid witness for the common input $x \in L$ chosen by P^* for that session. To be precise, P^* commits a valid witness w to c_w, as well as s_0 to c_{sk} (possibly by malleating verifier's public key into c_{sk}). Then, *possibly by malleating the first WIPOK sub-protocol concurrent interactions*, P^* successfully finishes the second WIPOK sub-protocol of the session by using the witness that, in turn, depends upon the secret key $SK \in \{s_0, s_1\}$ used by the honest verifier V. In more detail, when $SK = s_0$ (i.e., V uses s_0 as the witness in the first WIPOK sub-protocol interactions), it is the value committed to c_{sk}, i.e., s_0, that P^* uses as the witness in the second WIPOK sub-protocol of the session. On the other hand, when $SK = s_1$, it is the value committed to c_w, i.e., w, which could be maliciously related to s_1 as the common input x is selected by P^*. For this adversarial strategy, with non-negligible probability about $\frac{1}{2}$ (taken over the random bit $b \leftarrow \{0, 1\}$ where $SK = s_b$), the value extracted will just be the secret key that is also used by the extractor itself. However, we do not know how to reach a contradiction under standard polynomial-time assumptions in general. In particular, this adversarial strategy does not violate the statistical WI property of the first WIPOK sub-protocol: The values committed to (c_w, c_{sk}) are fixed, no matter which secret key is used by the verifier. Detailed clarification of this adversarial strategy (referred to as Adversarial Strategy 1 in p. 45), as well as that of some more exemplifying adversarial strategies, is given in Sect. 7.3.1.

[4] Strong WI (SWI) was first defined in [38], which actually refers to the issue that is fundamentally different from WI. Specifically, the issue is whether the interaction with the prover helps a malicious verifier V^* to distinguish some auxiliary information (which is indistinguishable without such an interaction). In particular, as already noticed in [38], SWI is *not* preserved under concurrent composition.

To overcome this technical difficulty, we employ complexity leveraging in a novel way. Specifically, on the system parameter n, we assume the OWF f is hard against sub-exponential 2^{n^c}-time adversaries for some constant c, $0 < c < 1$. However, the commitment c_{sk} is generated on a relatively smaller security parameter n_{sk} such that n_{sk} and n are polynomially related (i.e., any quantity that is a polynomial of n is also another polynomial of n_{sk} and vice versa) but $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$. This complexity leveraging ensures that, with at most negligible probability, the value extracted can be the secret key s_b committed to c_{sk}, from which the correctness of knowledge extraction (and then the *statistical* CKE security) is established. The reasoning is as follows. For any i, $1 \leq i \leq s(n)$, suppose that, with non-negligible probability p, an s-concurrent malicious P^* can successfully finish the ith session with c_{sk} committing to $s_\sigma, \sigma \in \{0, 1\}$, when the honest verifier (and also the extractor) uses s_σ as its secret key. Then, by the *statistical* WI property of the first WIPOK sub-protocol, with the same probability p, P^* successfully finishes the ith session with c_{sk} committing to s_σ, when the honest verifier uses $s_{1-\sigma}$ as the secret key. In the latter case, we can open c_{sk} to get s_σ by brute-force in $poly(n) \cdot 2^{n_{sk}}$-time, which, however, violates the sub-exponential hardness of y_σ because $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$.

We stress that complexity leveraging via the sub-exponential hardness assumption on verifier's public key is only for provable security analysis to frustrate concurrent man-in-the-middle. Both CZK simulation and CKE knowledge extraction are still in polynomial time (recall that we focus on black-box polynomial-time concurrent knowledge extraction in this work). We suggest that the use of complexity leveraging for frustrating CMIM adversaries could be a useful paradigm, different from the uses of complexity leveraging in existing works for protocols in the BPK model (e.g., [15,80]).

The CZK–CKE protocols from FSZK are roughly depicted in Fig. 1. Both the generic and efficient CZK–CKE protocols can be practically instantiated for some specific number-theoretic languages without going through general $\mathcal{NP}$-reductions. We also show that all other FSZK possible component variants within the given protocol structure of Fig. 1 are essentially not provably (black-box) CZK–CKE secure in the BPK model, which is, perhaps, somewhat puzzling. More details about the protocol constructions and security analysis of CZK–CKE in the BPK model are given in Sects. 6 and 7.

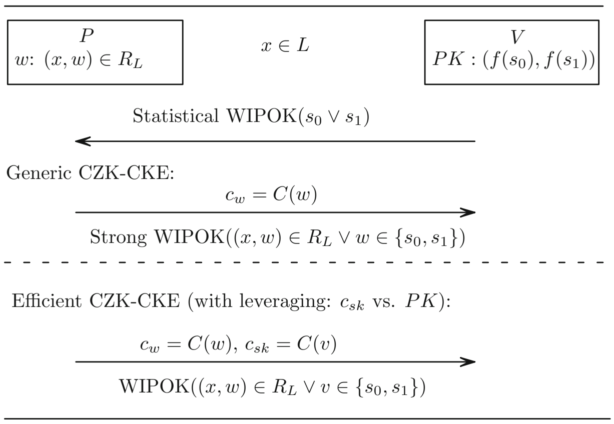

P
w: (x, w) ∈ R_L
x ∈ L
V
PK : (f(s_0), f(s_1))
Statistical WIPOK(s_0 ∨ s_1)
Generic CZK-CKE:
c_w = C(w)
Strong WIPOK((x, w) ∈ R_L ∨ w ∈ {s_0, s_1})
Efficient CZK-CKE (with leveraging: c_sk vs. PK):
c_w = C(w), c_sk = C(v)
WIPOK((x, w) ∈ R_L ∨ v ∈ {s_0, s_1})

Fig. 1. Depiction of CZK–CKE from FSZK .

1.2. *Related Works*

In general, the issue of concurrent composition of proof of knowledge could be traced back to the works [31,43]. The subtlety of argument of knowledge (AOK) in the BPK model was first observed in [29,30], where four distinct (namely one-time, sequential, concurrent and resettable) AOK notions were demonstrated with some unnatural protocols in the BPK model. By comparison, our separation between CKE and AOK in the BPK model is demonstrated with attacks on some naturally existing protocols (particularly, the fundamental FSZK protocol in the BPK model). Also, we separate CKE and concurrent soundness in the BPK model. The formulation approach of BPK concurrent AOK in [30] is significantly different from ours, with more details and discussions presented in Sect. 5.1.

Concurrent ZK (actually, resettable ZK that is stronger than CZK) arguments for $\mathcal{NP}$ with a *provable sub-exponential-time* CKE property in the BPK model were first achieved in [80], which, however, are available only for sub-exponentially hard languages and sub-exponentially hard verifier public keys. We note that the techniques used in [80] do not render CZK with *polynomial-time* concurrent knowledge extraction, and the subtle issues of knowledge extraction independence were not realized and formalized there.

Two constructions for concurrent ZK arguments with sequential soundness in the BPK model under standard assumptions were proposed in the incomplete work of [82]. However, the security proof of concurrent soundness turned out to be flawed, as observed independently in [28,81]. One construction was fixed to be concurrently sound in [28], and the other construction was fixed to be concurrently sound in [24] following the spirit of [28]. Given these works, the current work further shows that the concurrently sound CZK arguments of [24,28] do not capture CKE and are not concurrently knowledge extractable when it comes to proofs of knowledge.

Though we separate concurrent knowledge extraction from concurrent soundness and traditional argument of knowledge in the public-key model, to our knowledge the similar results for parallel/concurrent composition of proof/argument of knowledge in the plain model are unknown. For example, though the Goldreich–Kahan (GK) protocol [42] is constant-round ZK proof for $\mathcal{NP}$, it is still unknown whether the GK protocol is POK (in particular no attack is known to show that the GK protocol is indeed not POK). Based on the GK protocol, constant-round ZKPOK for $\mathcal{NP}$ is recently provably achieved in [62]. The notion of strong proof of knowledge (SPOK), where the knowledge extractor is required to run in strict polynomial time (rather than expected polynomial time), is introduced in [41] (see also [38]). The separation between SPOK and traditional POK (i.e., the existence of an interactive protocol that is POK but not SPOK) was conjectured in [41], which was later (partially) answered in [5,6]. The work [8] (see also [38]) discussed the difficulties with parallel repetition of proof of knowledge. Soundness error reduction under parallel repetition for argument systems, where the same argument protocol is run many times in parallel on the same fixed common input, were explored in [10,54,71].

Recently, the first constant-round simultaneously resettable zero-knowledge argument of knowledge in the BPK model under standard complexity assumptions was achieved by Cho, Ostrovsky, Scafuro and Visconti [17], which is referred to as the COSV protocol for presentation simplicity. At the heart of the COSV protocol is a celebrating building tool, i.e., constant-round simultaneously resettable witness-indistinguishable argument of knowledge (rWIAOK, for short) for $\mathcal{NP}$ in the plain model. The rWIAOK protocol is, in turn, based on the statistical ZKAOK protocol by Pass and Rosen [70]; as a consequence, the COSV protocol uses both *non-black-box* ZK simulation and *non-black-box* knowledge extraction. The COSV protocol is a strengthening of the (double-commitment based) efficient CZK–CKE protocol presented in Sect. 7 of this work, where the regular WIAOK in Stage 1 and Stage 3 is replaced by rWIAOK, which may, in some sense, further highlight the applicability of the efficient CZK–CKE protocol structure. The issue of CKE was not defined or discussed in [17], where AOK is relative to stand-alone protocol run. However, as to be discussed in Sect. 7.2 (p. 37) and Sect. 7.1 (p. 44), we observe that the COSV protocol seems to achieve *computational* (non-black-box) CKE security in the BPK model (the actual analysis is left for future work). In comparison with the possible rWIAOK-based solution, the efficient CZK–CKE protocol presented in this work enjoys (1) much more efficient computational and round complexity, as well as being able to be practically instantiated without going through $\mathcal{NP}$-reductions; (2) black-box ZK simulation and black-box knowledge extraction; and (3) statistical CKE.

A new model for round-efficient concurrent security, named the bounded player (BP) model, was recently introduced by Goyal, Jain, Ostrovsky, Richelson and Visconti [50]. The BP model is a further relaxation of the BPK model, as well as that of the bounded concurrency model [1]. In the BP model, the number of players is bounded; but unlike in the BPK model, there is no synchronization barrier between the key-generation stage and the proof stage. On the other hand, unlike in the bounded concurrency model, the number of sessions that may be involved is not *a priori* bounded. As discussed in [50], the BP model is strictly weaker than the BPK model. For example, while constant-round black-box CZK exists in the BPK model, non-black-box CZK simulation is necessary even for sub-logarithmic-round CZK in the BP model as achieved in [50]. Constant-round CZK in the BP model was recently achieved in [51], which might shed light on solving constant-round CZK in the plain model that is the central open problem in this area. But the issue of CKE was not considered in [50,51]. From our observations, the formulation and clarifications of CKE presented in this work may be extended or adapted to the BP model. We also observe that the protocols in [50,51] may not directly render the CKE security. The study of CZK–CKE in the BP model is left as an interesting future research direction.

1.3. *Organization*

We recall basic notions and tools in Sect. 2. In Sect. 3, we describe (an augmented version of) the BPK model with adaptive language selections based on public keys. In Sect. 4, we present the motivation, by concrete attacks on naturally existing protocol, for concurrent knowledge extractability in the public-key model. In Sect. 5, we formulate CKE in the BPK model and make clarifications and justification of the CKE formulation. In Sect. 6, we present the generic implementation of constant-round CZK–CKE arguments for $\mathcal{NP}$ in the BPK model under standard hardness assumptions. In Sect. 7, we present the efficient implementations of constant-round CZK–CKE arguments for $\mathcal{NP}$ in the BPK model with the usage of complexity leveraging in a minimal and novel way, and discuss and clarify in depth the various subtleties. In particular, we present and discuss the practical instantiations of both the generic and efficient CZK–CKE arguments for some number-theoretic languages without going through general $\mathcal{NP}$-reductions. We conclude this work by suggesting some open questions for future investigations in Sect. 8.

2. Preliminaries

In this section, we briefly recall some basic definitions and the corresponding constructions that are to be used in this work. The reader, who is familiar with these basics, can skip them (particularly, the constructions) on the first reading and will be referred back to them as needed in the subsequent sections.

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r. We let $y \leftarrow A(x_1, x_2, \ldots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots; r)$. If $\mathcal{S}$ is a finite set then $|\mathcal{S}|$ is its cardinality, and $x \leftarrow S$ is the operation of picking an element uniformly at random from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. We denote by $\mathbb{N}$ the set of natural numbers. By $[R_1; \ldots; R_n : v]$, we denote the set of values that a random variable v can assume, due to the distribution determined by the sequence of random processes $R_1, R_2, \ldots, R_n$. By $\Pr[R_1; \ldots; R_n : E]$, we denote the probability of event E, after the ordered execution of random processes $R_1, \ldots, R_n$. A string x is always assumed to be binary, and $|x|$ denotes its binary length.

It should be noted that, for presentation simplicity, the notation "$x_i \in \bar{x}$," where $\bar{x}$ is assumed to be the *vector* $(x_1, x_2, \ldots, x_k)$ is abused *informally* to emphasize the fact that x_i is the ith entry of the vector $\bar{x}$. For instance, the statement "for $x_i \in \bar{x}$," $1 \leq i \leq k$, is equivalent to "for the ith entry of $\bar{x}$." Throughout this work, unless otherwise specified, we denote by n (usually presented in unary as 1^n) the underlying security parameter.

Letting $\langle P, V \rangle$ 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 (resp., V) has private input x_1 (resp., x_2), and y_1 (resp., y_2) is the output of P (resp., V). We assume without loss of generality 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. When it comes to non-uniform security, we refer to non-uniform polynomial-time or sub-exponential-time algorithms.

Definition 2.1. *one-way function* A function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ is called a 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}, $\Pr[C(f(U_n), 1^n) \in f^{-1}(f(U_n))] < 2^{-n^c}$.

Definition 2.2. (*(public-coin) interactive argument/proof system* [14,48]) A pair of interactive machines, $\langle P, V\rangle$, is called an interactive argument system for a language L if both are PPT machines and the following conditions hold:

- Completeness. For every $x \in L \cap \{0,1\}^n$, there exists a string $w \in \{0,1\}^{poly(n)}$ such that for every string $z \in \{0,1\}^*$, $\Pr[\langle P(w), V(z)\rangle(x) = 1] = 1$.
- 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 $w, z \in \{0,1\}^*$, $\Pr[\langle P^*(w), V(z)\rangle(x) = 1]$ 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.

Commitment schemes enable a party, called the *sender*, to bind itself to a value in the initial *commitment* stage, while keeping the value secret against a potentially malicious *receiver* (this property is called *hiding*). Furthermore, when the commitment is opened by a potentially malicious sender in a later *decommitment* stage, it is guaranteed that the "opening" can yield only the single value determined in the commitment phase (this property is called *binding*). Commitment schemes come in two different flavors: statistically binding computationally hiding and statistically hiding computationally binding.

Definition 2.3. (*statistically/perfectly binding bit commitment scheme* [11,38]) A pair of PPT interactive machines, $\langle P, V\rangle$, is called a statistically/perfectly binding bit commitment scheme, if it satisfies the following:

Completeness. For any security parameter n, and any bit $b \in \{0,1\}$, it holds that

$$\Pr\left[\begin{array}{c}(\alpha, \beta) \leftarrow \langle P(b), V\rangle(1^n); \\ (t, (t, v)) \leftarrow \langle P(\alpha), V(\beta)\rangle(1^n)\end{array} : v = b\right] = 1.$$

Computationally hiding. For all sufficiently large n's, any PPT adversary V^*, the following two probability distributions are computationally indistinguishable:

$$[(\alpha, \beta) \leftarrow \langle P(0), V^*\rangle(1^n) : \beta], \text{ and}$$
$$[(\alpha', \beta') \leftarrow \langle P(1), V^*\rangle(1^n) : \beta'].$$

Statistically/perfectly binding. For all sufficiently large n's, and *any* (even computational power unbounded) adversary P^*, the following probability is negligible in n (or equals 0 for perfectly binding commitments):

$$\Pr\left[\begin{array}{c}(\alpha,\beta)\leftarrow\langle P^*,V\rangle(1^n);\\(t,(t,v))\leftarrow\langle P^*(\alpha),V(\beta)\rangle(1^n);\\(t',(t',v'))\leftarrow\langle P^*(\alpha),V(\beta)\rangle(1^n)\end{array}:v,v'\in\{0,1\}\bigwedge v\neq v'\right].$$

Below, we recall some classic perfectly binding commitment schemes.

One-round perfectly binding (computationally hiding) commitments can be based on any one-way permutation (OWP) [11,47]. Loosely speaking, given a OWP f with a hard-core predict b (cf. [38]), on the security parameter n one commits a bit σ by selecting $x \in \{0,1\}^n$ uniformly at random and sending $(f(x), b(x)\oplus\sigma)$ as a commitment, while keeping x as the decommitment information.

Let p and q be primes, $p = 2q + 1$ and $|p| = n$, and g be an element of $\mathbb{Z}_p^*$ of order q. We assume the decisional Diffie–Hellman (DDH) assumption holds on the cyclic group indexed by (p, q, g) (i.e., the sub-group of $\mathbb{Z}_p^*$ generated by g). For practical perfectly binding commitment scheme, in this work we use the DDH-based ElGamal (non-interactive) commitment scheme [34]. To commit to a value $v \in \mathbb{Z}_q$, the committer randomly selects $u, r \in \mathbb{Z}_q$, computes $h = g^u \mod p$ and sends $(h, \bar{g} = g^r, \bar{h} = g^v h^r)$ as the commitment. The decommitment information is (r, v). Upon receiving the commitment $(h, \bar{g}, \bar{h})$, the receiver checks that $h, \bar{g}, \bar{h}$ are elements of order q in $\mathbb{Z}_p^*$. It is easy to see that the commitment scheme is of perfectly binding. The computational hiding property is from the DDH assumption on the subgroup of order q of $\mathbb{Z}_p^*$ (for more details, see [34]). We also note that in [63] Micciancio and Petrank presented another implementation of DDH-based perfectly binding commitment scheme with advanced security properties.

Statistically binding commitments can be based on any one-way function (OWF) but run in two rounds [53,65]. On the security parameter n, letting $PRG : \{0,1\}^n \longrightarrow \{0,1\}^{3n}$ be a pseudorandom generator, the Naor's OWF-based 2-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 model.

Definition 2.4. (*trapdoor bit commitment scheme* [36]) A trapdoor bit commitment scheme (TC) is a quintuple of probabilistic polynomial-time algorithms TCGen, TCCom, TCVer, TCKeyVer and TCFake, such that

Completeness. For any security parameter n, and any bit $b \in \{0,1\}$, it holds that:

$$\Pr\left[\begin{array}{c}(TCPK,TCSK)\leftarrow\text{TCGen}(1^n);\\(c,d)\leftarrow\text{TCCom}(1^n,TCPK,b)\end{array}:\text{TCKeyVer}(1^n,TCPK)=\text{TCVer}(1^n,TCPK,c,b,d)=1\right]=1.$$

Computationally binding. For all sufficiently large n's and for any PPT adversary A, the following probability is negligible in n:

$$\Pr\left[\begin{array}{c}(TCPK, TCSK)\leftarrow \text{TCGen}(1^n);\\(c, v_1, v_2, d_1, d_2)\leftarrow \text{A}(1^n, TCPK)\end{array} : \begin{array}{c} v_1, v_2 \in \{0,1\} \bigwedge v_1 \neq v_2 \bigwedge \\ \text{TCVer}(1^n, TCPK, c, v_1, d_1) = \text{TCVer}(1^n, TCPK, c, v_2, d_2) = 1\end{array}\right].$$

Perfectly (or computationally) hiding. For all sufficiently large n's and any $TCPK$ such that $\text{TCKeyVer}(1^n, TCPK) = 1$, the following two probability distributions are identical (or computationally indistinguishable):

$$[(c_0, d_0) \leftarrow \text{TCCom}(1^n, TCPK, 0) : c_0], \text{ and } [(c_1, d_1) \leftarrow \text{TCCom}(1^n, TCPK, 1) : c_1].$$

Perfect (or computational) trapdoorness. For all sufficiently large n's and any $(TCPK, TCSK) \in \{\text{TCGen}(1^n)\}$, $\exists v_1 \in \{0, 1\}$, $\forall v_2 \in \{0, 1\}$ such that the following two probability distributions are identical (or computationally indistinguishable):

$$\left[\begin{array}{c}(c_1, d_1) \leftarrow \text{TCCom}(1^n, TCPK, v_1);\\ d_2' \leftarrow \text{TCFake}(1^n, TCPK, TCSK, c_1, v_1, d_1, v_2)\end{array} : (c_1, d_2')\right], \text{ and } [(c_2, d_2) \leftarrow \text{TCCom}(1^n, TCPK, v_2) : (c_2, d_2)].$$

Feige–Shamir trapdoor commitments (FSTC) [36]. Based on Blum's protocol for directed Hamiltonian cycle (DHC), Feige and Shamir developed a generic (computationally hiding and computationally binding) trapdoor commitment scheme [36], under any one-way permutation or any OWF (depending on the underlying perfectly binding commitment scheme used). The $TCPK$ of the FSTC scheme is $(y = f(x), G)$ (for OWF-based solution, $TCPK$ 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). The following is the description of the Feige–Shamir trapdoor bit commitment scheme, on the security parameter n.

Round-1. Letting 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 $3n$.

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 selects a random permutation, π, of the vertices V, and commits (using the underlying perfectly binding commitment scheme) the entries of the adjacency matrix of the resultant permutated graph. That is, it sends a q-by-q matrix of commitments so that the $(\pi(i), \pi(j))$-entry is a commitment to 1 if $(i, j) \in E$, or is a commitment to 0 otherwise. To commit to 1, the committer commits an adjacency matrix containing a randomly labeled 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 labeled cycle, and the receiver checks that all revealed values are 1 and the corresponding entries form a simple q-cycle.

Definition 2.5. (*witness indistinguishability WI* [37]) 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 $view_{V^*(z)}^{P(w)}(x)$ 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 $W^1 = \{w_x^1\}_{x\in L}$ and $W^2 = \{w_x^2\}_{x\in L}$ for sufficiently long x, so that $(x, w_x^1) \in R_L$ and $(x, w_x^2) \in R_L$, the following two probability distributions are computationally indistinguishable by any polynomial-time algorithm:

$$\left\{view_{V^*(z)}^{P(w_x^1)}(x)\right\}_{x\in L,\, z\in\{0,1\}^*}, \text{ and}$$
$$\left\{view_{V^*(z)}^{P(w_x^2)}(x)\right\}_{x\in L,\, z\in\{0,1\}^*}.$$

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

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

Definition 2.6. (*strong witness indistinguishability SWI* [38]) Let $\langle P, V\rangle$ and all other notations be as in Definition 2.5. We say that $\langle P, V\rangle$ is *strongly witness indistinguishable for* R_L if for every PPT interactive machine V^* and for every two probability ensembles $\{(X_n^1, Y_n^1, Z_n^1)\}_{n\in\mathbb{N}}$ and $\{(X_n^2, Y_n^2, Z_n^2)\}_{n\in\mathbb{N}}$, such that each (X_n^i, Y_n^i, Z_n^i) ranges over $(R_L \times \{0, 1\}^*) \cap (\{0, 1\}^n \times \{0, 1\}^* \times \{0, 1\}^*)$, the following holds: If $\{(X_n^1, Z_n^1)\}_{n\in\mathbb{N}}$ and $\{(X_n^2, Z_n^2)\}_{n\in\mathbb{N}}$ are computationally indistinguishable, then so are $\{\langle P(Y_n^1), V^*(Z_n^1)\rangle(X_n^1)\}_{n\in\mathbb{N}}$ and $\{\langle P(Y_n^2), V^*(Z_n^2)\rangle(X_n^2)\}_{n\in\mathbb{N}}$.

WI versus SWI: It is clarified in [39] that the notion of SWI actually refers to issues that are fundamentally different from WI. Specifically, the issue is whether the interaction with the prover helps V^* to distinguish some auxiliary information (which is indistinguishable without such an interaction). Significantly different from WI, SWI does *not* preserve under concurrent composition. For more detailed clarifications and discussions about SWI, the reader is referred to [39]. However, an interesting observation (as shown in Sect. 6) is: the protocol composing commitments and SWI can be itself regular WI.

Definition 2.7. (*system for argument/proof of knowledge* [9,38]) Let R be a binary relation and $\kappa : \mathbb{N} \to [0, 1]$. We say that a probabilistic polynomial-time 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 deterministic 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 argument/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 argument/proof of knowledge (AOK/POK) for the relation R.

The above definition of POK is with respect to *deterministic* prover strategy. POK also can be defined with respect to *probabilistic* prover strategy. It is shown that the two definitions are equivalent for all natural cases (e.g., POK for $\mathcal{NP}$-relations) [9].

We mention that Blum's protocol for DHC [12] is just a 3-round public-coin WIPOK for $\mathcal{NP}$, which is recalled below.

Blum's protocol for DHC [12]. The n-parallel repetitions of Blum's basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [12] 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 2-round (public-coin) perfectly binding commitment scheme [65]. 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 $(\pi(i), \pi(j))$-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 DHC relies on the hiding property of the underlying perfectly binding commitment scheme used in its first round.

The Lapidot–Shamir protocol (LS protocol) for DHC [59]. The n-parallel repetition of the Lapidot–Shamir basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [59] is another 3-round public-coin WIPOK for $\mathcal{NP}$ (with knowledge error 2^{-n}) under any one-way permutation (as its first round 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 2-round (public-coin) statistically binding commitment scheme [65]. The following is the description of the Lapidot–Shamir *basic* protocol for DHC (that is also described in [35]):

Round- 1. The prover P commits an adjacency matrix for a randomly labeled 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).

One salient feature of the LS protocol is that the prover can send 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 when 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 the LS protocol for DHC relies on the hiding property of the underlying perfectly binding commitment scheme (used in its first-round).

Statistical WI argument/proof of knowledge (WIA/POK). We employ, in a critical way, constant-round *statistical* WIA/POK in this work. We briefly mention two simple ways for achieving constant-round statistical WIA/POK systems. Firstly, for any statistical/perfect Σ protocol (cf. Sect. 2.1), the OR proof (i.e., the Σ_{OR} protocol) is statistical/perfect WI proof of knowledge. The second approach is to modify the (parallel repetition of) Blum's protocol for DHC [12] (that is computational WIPOK) into constant-round statistical WIAOK by replacing the statistically binding commitments (used in the first round of Blum's protocol) with constant-round *statistically hiding* commitments. One-round statistically hiding commitments can be based on any collision-resistant hash function [23,57]. Two-round statistically hiding commitments can be based on any claw-free collection with an efficiently recognizable index set [38,42,49], or based on any collision-resistant hash function (CRHF) [23,57].[5] Statistically hiding commitments can also be based on general assumptions, in particular any OWF, but with non-constant rounds [55,56,66]).

Argument/proof of knowledge (A/POK) as sub-protocols [61]. Arguments or proofs of knowledge are often used as sub-protocols within larger protocols. In order to simulate the rest of the larger protocol subsequent to a successful run of the A/POK sub-protocol, a simulator needs to run the extractor for the A/POK to obtain some secret information. This secret information is then used in order to simulate the rest of the larger protocol.

However, a technical problem arises when using an A/POK sub-protocol in such a way. Specifically, if the knowledge error κ (as defined in Definition 2.7) is negligible, the running time of the simulator is not guaranteed to be expected polynomial time. This technical difficulty was solved by Goldreich and Kahan in [42]. However, the techniques proposed in [42] are complex, and thus, applying them every time, we need to use an A/POK sub-protocol is cumbersome. To simplified the matter and to ease modular use and analysis of A/POK as sub-protocols, Lindell [61] introduces the notion of witness-extended emulator and presents a general lemma that enables the use of A/POK as sub-protocols without requiring any complicated analysis.

Definition 2.8. (*witness-extended emulator* [61]) Let R be a binary relation, and let $\langle P, V\rangle$ be an interactive proof system. Consider a probabilistic expected polynomial-time machine $E = (E_1, E_2)$ that is given input x and access to the oracle $P^*_{x,w,r}$, where $P^*_{x,w,r}$ denotes the strategy of the (possibly malicious) prover P^* upon common input x, auxiliary input w such that $(x, w) \in R$, and random tape r. We denote by $view_V^{P^*_{x,w,r}}(x)$ a random variable describing the view of the honest verifier V in an execution of the A/POK with $P^*_{x,w,r}$ (which consists of the random coins of V and the messages received from $P^*_{x,w,r}$). Let $E_1^{P^*_{x,w,r}}(x)$ and $E_2^{P^*_{x,w,r}}(x)$ denote the random variables representing the first and second elements of the output of E, respectively. We say that E is a witness-

[5] The CRHF-based statistically hiding commitment scheme can be non-interactive, but is only secure against uniform adversaries or is secure in the sense of "human ignorance" [74].

extended emulator[6] for $\langle P, V\rangle$ and R, if for every interactive function P^*, every $x, w, r \in \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large x's,

1. E_1 outputs the distribution of V's view in a real execution with $P^*_{x,w,r}$. Specifically, the following probability ensembles are identical: $\left\{E_1^{P^*_{x,w,r}}(x)\right\}_{x,w,r}$ and $\left\{view_V^{P^*_{x,w,r}}\right\}_{x,w,r}$.
2. The probability that V's view (as output by E_1) is accepting, and yet E_2 does not output a correct witness $\hat{w}$ such that $(x, \hat{w}) \in R$ is negligible (in $|x|$).

Lemma 2.1. (witness-extended emulation lemma [61]) *Let R be a binary relation and let $\langle P, V\rangle$ be an argument/proof of knowledge for R with negligible knowledge error. Then, there exists a witness-extended emulator $E = (E_1, E_2)$ for $\langle P, V\rangle$ and R.*

2.1. Σ *and* Σ_{OR} *Protocols*

Σ protocols are 3-round public-coin protocols satisfying a special honest-verifier zero-knowledge (SHVZK) property and a special soundness property in the sense of knowledge extraction.

Definition 2.9. (Σ *protocol* [19]) A 3-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ protocol for an $\mathcal{NP}$-language with relation R_L if the followings hold:

- Completeness. If P, V follow the protocol, the verifier always accepts.
- Special soundness. From any common input x of length $poly(n)$ 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 $(x, w) \in R_L$. Here a, e and z stand for the first-, the second- and the third-round message, respectively and e is assumed to be a string of length k (such that k is polynomially related to the security parameter n) selected uniformly at random in $\{0,1\}^k$.
- Special honest verifier zero knowledge (SHVZK). There exists a probabilistic polynomial-time simulator S, which on input x (where there exists a w such that $(x, w) \in R_L$), and a random challenge string $\hat{e}$ outputs an accepting conversation of the form $(\hat{a}, \hat{e}, \hat{z})$, with the probability distribution that is indistinguishable from that of the real conversation (a, e, z) between the honest $P(w)$ and V on input x.

The first Σ protocol (for an $\mathcal{NP}$-language) in the literature can be traced back to the GMW protocol for graph isomorphism [47], though the name of Σ protocol is only adopted later in [19] (we note that the GMW protocol for G3C [47] is 3-round public-coin SHVZK but not of special soundness, and thus not a Σ protocol by definition). A Σ protocol is called *perfect/statistical* Σ protocol, if it is perfect/statistical SHVZK. A Σ protocol is called *partial witness-independent*, if the generation of its first-round message from prover is independent of (i.e., without using) the witness for the common input. Since [47], a very large number of Σ protocols have been developed in the literature.

[6] The machine E is named a "witness-extended emulator" due to the fact that the main goal is *emulation* while the witness extraction is a tool used to accomplish this goal [61].

In particular, (the n-parallel repetition of) Blum's protocol for DHC [12] is a (partial witness-independent) computational Σ protocol for $\mathcal{NP}$; that is, the n-parallel repetition of Blum's protocol for DHC [12] is also a 3-round (partial witness-independent) WI for $\mathcal{NP}$. Most practical Σ protocols for number-theoretic languages (e.g., DLP and RSA [52,76]) are (partial witness-independent) *perfect* Σ protocols. For a good survey of Σ protocols and their applications, the reader is referred to [22].

Σ-Protocol for DLP [76]. The following is a Σ protocol $\langle P, V\rangle$ proposed by Schnorr [76] 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 the 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 $\mathbb{Z}_p^*$ of order q.

- In the first round, P chooses r at random in $\mathbb{Z}_q$ and sends $a = g^r \; mod \; p$ to V.
- In the second round, V chooses a challenge e at random in $\mathbb{Z}_{2^k}$ and sends it to P. Here, k is fixed such that $2^k < q$.
- In the third round, P sends $z = r + ew \; mod \; q$ to V, who checks that $g^z = ah^e \; mod \; p$, that p and q are primes and that g and h have order q, and accepts iff this is the case.

Σ protocol for the q-th root problem (QRP) [52]. Let N be an RSA modulus (i.e., the product of two large primes), and let $q < N$ be a prime. Given a value $y \in \mathbb{Z}_N$, the qth root problem (QRP) over $\mathbb{Z}_N^*$ is to find an element $x \in \mathbb{Z}_N$ such that $y = x^q \mod N$. The QRP problem is assumed to be one-way (without knowing the order of $\mathbb{Z}_N^*$), where the probability is taken over the random choices of (N, x, y) and the random coins of the attacker. Note that, if $q = 2$, the QRP problem is just the square root problem (modulo N) whose hardness is computationally equivalent to that of factoring N. For $q > 2$, the QRP problem can be viewed as a special case of the RSA problem, which is conjectured to be easier than factoring [13].

Let $U = (N, q, y)$ be the common input, and $w \in \mathbb{Z}_N^*$ be the private input, where $y = w^q \mod N$. The following is a Σ protocol for the QRP problem [52]:

- P chooses r at random in $\mathbb{Z}_N^*$ and sends $a = r^q \mod N$ to V.
- V chooses a challenges e uniformly at random in $\{0,1\}^l$ and sends e to P, where l is fixed such that $2^l < q$.
- P sends $z = rw^e \mod N$ to V, who checks that $a = z^q/y^e \mod N$, 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 [20]. One basic construction with Σ protocols is the OR of a real protocol conversation and a simulated one, called Σ_{OR}, that allows a prover to show that given two inputs x_0, x_1 (for possibly different $\mathcal{NP}$-relations R_0 and R_1, respectively), it knows a w such that either $(x_0, w) \in R_0$ or $(x_1, w) \in R_1$, without revealing which is the case (i.e., witness-indistinguishable WI) [20]. 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 k, consider the following protocol $\langle P, V\rangle$, which we call Σ_{OR} protocol. 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 lets $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output. P finally sends a_0, a_1 to V.
- V chooses a random k-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 2.1. [20] *The protocol Σ_{OR} above is a Σ protocol for R_{OR}, where $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R_0 \text{ or } (x_1, w) \in R_1\}$. Moreover, Σ_{OR} protocols are witness-indistinguishable (WI) argument or proof of knowledge systems.*

The SHVZK simulator of Σ_{OR} [20]. For a Σ_{OR} protocol of the above form, denote by S_{OR} the SHVZK simulator of it, and denote by S_b the SHVZK simulator of the protocol $\langle P_b, V_b\rangle$ for $b \in \{0, 1\}$. Then on common input (x_0, x_1) and a random string $\hat{e}$ of length k, $S_{OR}((x_0, x_1), \hat{e})$ works: It first chooses a random k-bit string $\hat{e}_0$, computes $\hat{e}_1 = \hat{e} \oplus \hat{e}_0$; then, S_{OR} runs $S_b(x_b, \hat{e}_b)$ to get a simulated transcript $(\hat{a}_b, \hat{e}_b, \hat{z}_b)$ for $b \in \{0, 1\}$; finally, S_{OR} outputs $((\hat{a}_0, \hat{a}_1), \hat{e}, (\hat{e}_0, \hat{z}_0, \hat{e}_1, \hat{z}_1))$.

3. The Bare Public-Key Model

We present the definitions of concurrent soundness and concurrent zero knowledge in the BPK model (cf. [15,64]). The key augmentation with the current formulation, in comparison with the previous definition of the BPK model, is to explicitly consider adaptive language and statement selection based on public keys.

3.1. *Honest Players in the BPK Model*

Let R_{KEY} be an $\mathcal{NP}$-relation validating the public key and secret key pair (PK, SK) generated by honest verifiers in the BPK model, i.e., $R_{KEY}(PK, SK) = 1$ indicates that SK is a valid secret key of PK. In this work, the relation R_{KEY} implicitly determines a one-way function, denoted f_{KEY}, such that the ability of computing SK from PK implies breaking the one-wayness of f_{KEY}. We assume R_{KEY}, as well as f_{KEY}, is implicitly specified in PK; for presentation simplicity, it may be omitted in protocol specifications.

A protocol $\langle P, V\rangle$ in the BPK model on a security parameter 1^n, w.r.t. some key-validating relation R_{KEY} and some $\mathcal{NP}$-relation R_L for an $\mathcal{NP}$-language L, consists of the following:

- F: 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. When verifier's IDs are implicitly specified from the context, for presentation simplicity, we also just take F as a collection of public keys in protocol specification and security analysis.

- $P(1^n, R_L, x, w, F, id, \gamma)$: An honest prover that is a polynomial-time interactive machine, where 1^n is the security parameter, x is a $poly(n)$-bit string in L, w is a witness such that $(x, w) \in R_L$, F is a public file, id is a verifier identity, and γ is its random tape.
- V: An honest verifier that is a polynomial-time interactive machine, working in two stages.
  1. Key-generation stage: V, on the security parameter 1^n and a random tape r, outputs a key pair (PK, SK) satisfying $R_{KEY}(PK, SK) = 1$. V then registers PK in F as its public key while keeping the corresponding secret key SK in private.
  2. Proof stage: V, on input SK and R_L, $x \in \{0, 1\}^{poly(n)}$ (which is supposed to be in L) and a random tape ρ, performs an interactive protocol with a prover and outputs "accept" indicating $x \in L$ or "reject" indicating $x \notin L$.

Some remarks about our formulation of the BPK model are in order. For an interactive protocol in the BPK model, as we shall show in this work, whether or not the underlying language L being set based on the public keys makes a distinction. Specifically, there are two cases to consider. One case is that the language L is *a priori* fixed before honest verifiers generate public keys. The other case is that the language is set only after public keys are registered, and in this case, the language may be dependent upon the public keys (particularly the key-validating relation R_{KEY} and the underlying OWF f_{KEY}). For the latter case, as we shall show in Sect. 4.2, concurrent soundness does not ensure concurrent verifier security in public-key model. To our knowledge, this issue was not made clear in the literature. In addition, as players' public keys are static and are usually applied across different cryptographic applications, the static public keys can be viewed as part of the common input for interactive systems in public-key model. In this sense, for a protocol $\langle P, V\rangle$ in the BPK model, the time complexity of an interactive machine depends both upon the underlying language to be proved and upon the registered public keys (in particular, R_{KEY} and f_{KEY}). For example, suppose the public keys can be generated with any one-way function f_{KEY}. Then, for different instantiations of the OWF f_{KEY}, the time complexity of P or V can be different.[7]

In our formulation, an interactive protocol in the BPK model is specified w.r.t. both the key-validating relation R_{KEY} and the underlying language L. Moreover, the underlying language L, w.r.t. which the protocol will be conducted, can be dependent upon the static public keys of the honest verifiers (particularly, the relation R_{KEY} and the underlying OWF f_{KEY}). For example, R_L may comprise, or just be equal to, R_{KEY}. We suggest that this is realistic in practice for cryptographic protocols running concurrently in the public-key model, where cryptographic applications, and hence, the underlying languages or statements usually depend upon players' static public keys.

[7] For standard definition of interactive protocol in the plain model [38,48], a "polynomial-time" interactive machine means, roughly speaking, that its running time is $p(|x|)$ for sufficiently large common input $x \in \{0, 1\}^*$, where p is some fixed polynomial. The time complexity is defined implicitly w.r.t. a specific language L alone.

3.2. *The Malicious Concurrent Prover and Concurrent Soundness in BPK Model*

Let (PK, SK) be the output of the key-generation stage of V on the security parameter 1^n under the key-validating relation R_{KEY}, and R_L the $\mathcal{NP}$-relation for a language L *that may possibly depend upon* R_{KEY}. An s-concurrent malicious prover P^* in the BPK model, for a positive polynomial s is a probabilistic polynomial-time turing machine that, on the security parameter 1^n, PK, R_L and an auxiliary string $z \in \{0,1\}^*$, performs an s-concurrent attack against V as follows.

P^* can perform concurrently at most $s(n)$ interactive protocols (sessions) with (the proof stage of) V as follows: If P^* is already running $i-1$ $(1 \leq i \leq s(n))$ 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 proof stage of $V(1^n, R_L, x_i, SK, \rho_i)$. P^* can output a message for any running protocol and always promptly receive the response from V. That is, P^* controls at its will 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 proof stage (that is, $\rho_1, \ldots, \rho_{s(n)}$ are independent random strings). We denote by $view_{P^*}^{V(SK)}(1^n, PK, z)$, where $(PK, SK) \in R_{KEY}$ are generated by V on the security parameter 1^n, the random variable describing the view of P^* in this experiment, which consists of its random tape, the auxiliary string z, all messages it receives including the public key PK and all messages sent by $V(1^n, R_L, x_i, SK, \rho_i)$'s in the $s(n)$ proof stages, $1 \leq i \leq s(n)$.

We then say a protocol $\langle P, V \rangle$, which is w.r.t. some key-validating relation R_{KEY} and some $\mathcal{NP}$-relation R_L, is *concurrently sound* in the BPK model, if for any sufficiently large n, for any honest verifier V and all, except for a negligible fraction of, (PK, SK) output by the key-generation stage of V under R_{KEY}, for any positive polynomial s and any s-concurrent malicious prover P^* and any string $z \in \{0,1\}^*$, for any string $x \notin L$ of length $poly(n)$, the probability that $V(1^n, R_L, SK)$ outputs "accept $x \in L$" in one of the $s(n)$ sessions is negligible in n, where the probability is taken over the randomness of P^*, the randomness of V for key generations and for all the $s(n)$ proof stages.

The above concurrent soundness is defined w.r.t multiple proof stages (sessions) with the same public key. In this case, we can assume that the auxiliary information z encodes information collected from protocol executions w.r.t. other public keys that are generated independently of the public key PK at hand. Note that, as discussed in [64], extension to the general case, where P^* interacts with instances of multiple verifiers with multiple (independently generated) public keys, is direct.

3.3. *The Malicious Concurrent Verifier and Concurrent ZK in the BPK Model*

An s-concurrent malicious verifier V^*, where s is a positive polynomial, is a PPT turing machine that, on input 1^n and an auxiliary string $z \in \{0,1\}^*$, works in two stages:

Stage-1 (key-generation stage). On $(1^n, z)$, V^* outputs an arbitrary public file F and a list of $s(n)$ identities $id_1, \ldots, id_{s(n)}$. Then, V^* is given a list of $s(n)$ strings $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$ of length $poly(n)$ each, where x_i might be equal to x_j, $1 \leq i, j \leq s(n)$.

Stage-2 (proof stage). Starting from the final configuration of Stage 1, V^* concurrently interacts with $s(n)^2$ instances of the honest prover P: $P(1^n, F, R_L, x_i, w_i,$

$id_j, \gamma_{(i,j)})$, where $1 \leq i, j \leq s(n)$, $(x_i, w_i) \in R_L$ and $\gamma_{(i,j)}$'s are independent random strings. In this stage, V^* controls at its will the schedule of the messages being exchanged in all the concurrent sessions. In particular, V^* can output a message for any running session dynamically based on the transcript up to now and always promptly receive the response from P.

For any auxiliary string $z \in \{0,1\}^*$, any public-key file F output by $V^*(1^n, z)$ in Stage 1 and any $\bar{\mathbf{x}} = \left(x_1, \ldots, x_{s(n)}\right) \in L^{s(n)}$, we denote by $view_{V^*(z)}^{\{P(F, R_L, x_i, w_i, id_j, \gamma_{(i,j)})'s\}}(1^n, \bar{\mathbf{x}})$ the random variable describing the view of V^* in its second stage of this experiment, which consists of $(z, F, R_L, \bar{\mathbf{x}})$, the randomness of V^* in its second stage, and all messages received from all the $s(n)^2$ prover instances.

Definition 3.1. *(concurrent zero knowledge in the BPK model)* A protocol $\langle P, V \rangle$ for a language L with $\mathcal{NP}$-relation R_L is (black-box) concurrent zero knowledge in the BPK model, if there exists a PPT black-box simulator S such that for any sufficiently large n and every s-concurrent malicious verifier V^* the following two distribution ensembles are indistinguishable:

$$\left\{view_{V^*(z)}^{\{P(1^n, F, R_L, x_i, w_i, id_j, \gamma_{(i,j)})'s\}}(1^n, \bar{\mathbf{x}})\right\}_{\bar{\mathbf{x}} \in L^{s(n)}, F \in \{0,1\}^*, z \in \{0,1\}^*}, \text{ and}$$
$$\left\{S(1^n, F, R_L, \bar{\mathbf{x}}, z)\right\}_{\bar{\mathbf{x}} \in L^{s(n)}, F \in \{0,1\}^*, z \in \{0,1\}^*}.$$

4. Motivation for Concurrent Knowledge Extraction in the Public-Key Model

In this section, by concrete concurrent interleaving and malleating attacks on some natural ZK protocols in the public-key setting, we demonstrate that both argument of knowledge and concurrent soundness do not guarantee concurrent verifier security (i.e., fail to ensure knowledge extraction in concurrent executions) in the public-key model. These concrete attacks serve as a good motivation for understanding "possession of knowledge on the Internet with registered public keys," i.e., the subtleties of concurrent knowledge extraction (CKE) in the public-key model.

4.1. *AOK Does Not Ensure CKE in the Public-Key Model*

We demonstrate this by a concrete attack on the basic Feige–Shamir zero-knowledge (FSZK) protocol [35,36] in the public-key model. The FSZK protocol is conceptually simple, which is simply composed of two WIPOK sub-protocols. In more detail, letting f be a OWF, in the first WIPOK sub-protocol with the verifier V serving as the knowledge prover, V computes $(y_0 = f(s_0), y_1 = f(s_1))$ for randomly chosen s_0 and s_1; then, V proves to the prover P the knowledge of the preimage of either y_0 or y_1. In the second WIPOK sub-protocol with P serving as the knowledge prover, on common input x, P proves to V the knowledge of either a valid $\mathcal{NP}$-witness w for $x \in L$ or the preimage of either y_0 or y_1. FSZK is zero-knowledge argument of knowledge and can be practically instantiated (without going through general $\mathcal{NP}$-reductions) by implementing the underlying WIPOK sub-protocols with the Σ_{OR} technique [20].

Though FSZK was originally proposed in the plain model [35,36], it may appear quite natural for the verifier to publish (y_0, y_1) as its public key and set s_b (for a random bit b) as its secret key when deploying FSZK in practice [82]. This public-key version of FSZK has indeed served as the basis for a list of round-efficient concurrent and resettable ZK protocols in the literature[17,24,25,28,68,75,77,79,81,82]. However, the following attack (that was originally presented in [81] and is now combined into this work), on the Σ_{OR}-based implementation of FSZK (with the underlying WIPOK implemented by Σ_{OR}) in the public-key model, shows that this intuition is wrong.

Let L (WLOG, the $\mathcal{NP}$-complete language directed Hamiltonian cycle (DHC)) be a language that admits Σ protocols. We show how a malicious prover P^* can convince an honest verifier V (with public key (y_0, y_1)) of a statement "$x \in L$" *without possessing any witness to*"$x \in L$," by concurrently interacting two sessions with V. The message schedule of P^* in the two sessions is specified as follows, which is also diagrammatically presented in Fig. 2.

1. P^* interacts with V in the first session and works just as the honest prover does in the first Σ_{OR} protocol (i.e., the first WIPOK sub-protocol).
2. When P^* moves into the second Σ_{OR} protocol of the first session, and needs to send V the first-round message, denoted a_P, of the second Σ_{OR} protocol of this session on common input (x, y_0, y_1), P^* suspends the first session and does the following:
   - It first runs the SHVZK simulator (of the underlying Σ protocol for L) on x to get a simulated conversation, denoted (a_x, e_x, z_x), for the statement "$x \in L$."
   - Then, P^* initiates the second session with V. After receiving the first-round message, denoted a'_V, of the first Σ_{OR} protocol in the second session on common input (y_0, y_1) (i.e., V's public key), P^* sets $a_P = (a_x, a'_V)$ and suspends the second session.

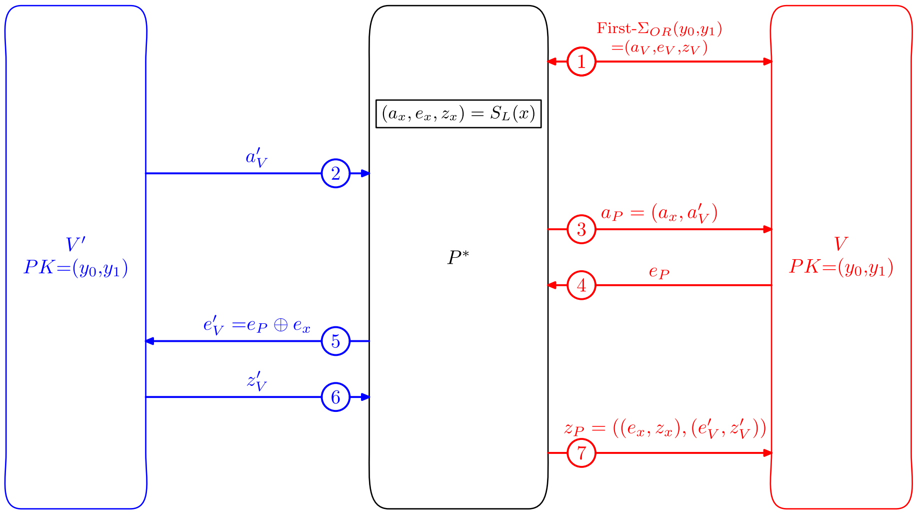


Fig. 2. Concurrent interleaving attack on FSZK in public-key model .

3. 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 second Σ_{OR} protocol in the first session.
4. P^* runs V further in the first session. After receiving the second-round message of the second Σ_{OR} protocol of the first session, denoted by e_P (i.e., the random challenge from V), P^* sets $e'_V = e_P \oplus e_x$ and suspends the first session again.
5. P^* continues the execution of the second session and sends $e'_V = e_P \oplus e_x$ to V as its random challenge in the second-round of the first Σ_{OR} protocol in the second session.
6. After receiving the third-round message of the first Σ_{OR} protocol in the second session, denoted z'_V, P^* sets $z_P = ((e_x, z_x), (e'_V, z'_V))$ and suspends the second session again.
7. P^* continues the execution of the first session again, sending the value $z_P = ((e_x, z_x), (e'_V, z'_V))$ to V as the last-round message of the first session.

Note that (a_x, e_x, z_x) is an accepting conversation for showing "$x \in L$," and (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_0 or y_1, and furthermore, $e_P = e_x \oplus e'_V$. According to the description of Σ_{OR} (cf. Sect. 2.1), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation on common input (x, y_0, y_1) for the second Σ_{OR} protocol in the first session, and thus, P^* successfully convinced V of the statement "$x \in L$" in the first session *without knowing any witness to it*. Moreover, the statement "$x \in L$" can even be false.

4.2. *Concurrent Soundness Does Not Ensure CKE in the Public-Key Model*

We next show a concurrent interleaving and malleating attack on the concurrent ZK protocol of [28,82] that is *concurrently sound* in the BPK model. The attack is demonstrated w.r.t. a language (set by a malicious prover P^*) that is dependent on verifier's public key. Similar to the attack against FSZK in the public-key model, by concurrently interacting with the honest verifier in two sessions, the malicious prover P^* can successfully (with probability 1) malleate the verifier's interactions in one session into successful interactions in another session on a *true* (public-key-related) statement *without knowing any witness to the statement being proved.* We remark that the concurrent soundness property of the CZK protocol of [28] still holds, even though the malicious prover can set the language and statements (to be proved) dependent upon verifiers' public keys. This shows a gap between concurrent soundness and concurrent knowledge extraction in the public-key model, particularly when the language and statements (set by a malicious prover) may be dependent on verifiers' public keys. Actually, as we shall prove in Sect. 5, CKE is *strictly* stronger than concurrent soundness in the public-key model assuming the existence of OWF.

4.2.1. *The Protocol Structure of [31,82]*

Key generation. Let f be a OWF that admits Σ protocols. On the security parameter n, each verifier V randomly selects two elements in the domain of f, s_0 and s_1 of length n each and computes $y_0 = f(s_0)$ and $y_1 = f(s_1)$. V publishes (y_0, y_1) as its public key while keeping s_b as its secret key for a randomly chosen b from $\{0, 1\}$

(For OWF-based implementation, V also publishes a random string r_V of length $3n$ that serves as the first-round message of Naor's OWF-based perfectly binding commitment scheme [65].).

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

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

Phase- 1. The verifier V proves to P that it knows the preimage of either y_0 or y_1, by executing the Σ_{OR} protocol on (y_0, y_1) in which V plays the role of the knowledge prover. It is additionally required that the first-round message of the Σ_{OR} protocol be generated without using the preimage of either y_0 or y_1 (i.e., *partial witness-independent*). Denote by a_V, e_V and z_V, the first-, the second- 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 (For OWF-based implementation, P sends a random string r_P of length $3n$ on the top, which serves as the first-round message of Naor's OWF-based perfectly binding commitments and is used by V in generating a_V.).

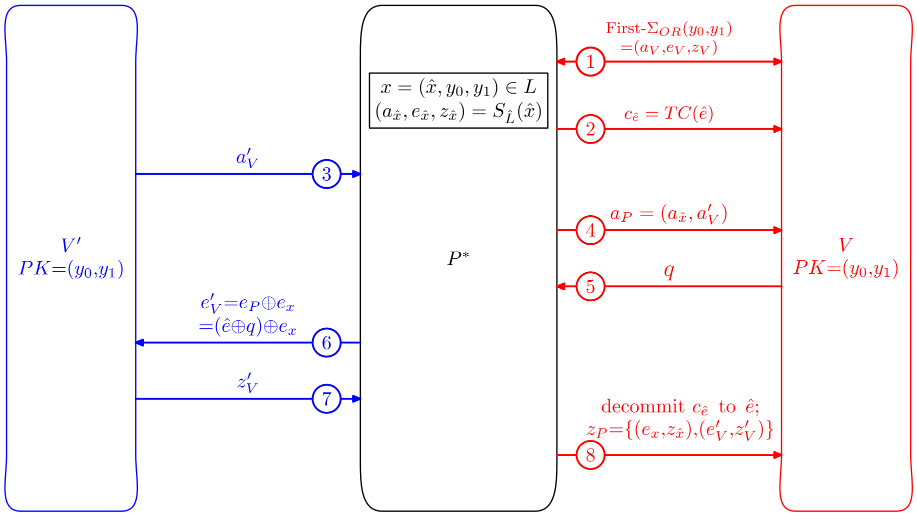
If V successfully finishes the Σ_{OR} protocol of this phase and P accepts, then go to Phase 2. Otherwise, P aborts.

Phase- 2. Let TC be a trapdoor bit commitment scheme with the preimage of either y_0 or y_1 as the trapdoor. The prover randomly selects a string $\hat{e} \in \{0, 1\}^n$ and sends $c_{\hat{e}} = \{TCCom(\hat{e}_1), TCCom(\hat{e}_2), \ldots, TCCom(\hat{e}_n)\}$ to the verifier V, where $\hat{e}_i$ is the ith bit of $\hat{e}$.

Phase- 3. Phase 3 runs essentially the underlying Σ protocol for L, except that the random challenge now 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 (for OWF-based implementation, a_P is also computed using r_V published by V in the key-generation phase); 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 $x \in L$.

Remark. The above protocol structure is essentially that of the CZK protocol of [82] (cf. Figure-3 of [82]) and can be implemented based on any OWF. The key difference in the actual implementations of [28,82] is that the work [28] uses a special trapdoor commitment scheme in Phase 2, where the decommitment information to 0 or 1 is in turn committed to two statistically binding commitments, which is critical for achieving concurrent soundness. We remark that the differences in actual implementations do not invalidate the attack presented below in Sect. 4.2.2, which is presented with respect to a more general protocol structure. For presentation simplicity, we refer to the protocol structure from [28,82] as DVZK.



V'
$PK=(y_0,y_1)$
$x=(\hat{x},y_0,y_1)\in L$
$(a_{\hat{x}},e_{\hat{x}},z_{\hat{x}})=S_{\hat{L}}(\hat{x})$
P^*
V
$PK=(y_0,y_1)$
1 First-$\Sigma_{OR}(y_0,y_1)=(a_V,e_V,z_V)$
2 $c_{\hat{e}}=TC(\hat{e})$
3 a'_V
4 $a_P=(a_{\hat{x}},a'_V)$
5 q
6 $e'_V=e_P\oplus e_x=(\hat{e}\oplus q)\oplus e_x$
7 z'_V
8 decommit $c_{\hat{e}}$ to $\hat{e}$; $z_P=\{(e_x,z_{\hat{x}}),(e'_V,z'_V)\}$

Fig. 3. Concurrent interleaving attack on DVZK with PK-related language .

4.2.2. *The Concurrent Interleaving and Malleating Attack*

With respect to the above protocol structure of the protocols of [28,82], let $\hat{L}$ be any $\mathcal{NP}$-language admitting a Σ protocol that is denoted by $\Sigma_{\hat{L}}$ (*in particular*, $\hat{L}$ *can be an empty set*). Then for an honest verifier V with its public key $PK = (y_0, y_1)$, we define a new language $L = \{(\hat{x}, \hat{y}_0, \hat{y}_1) | \exists (w, b) \in \{0, 1\}^* \times \{0, 1\}\ s.t.:\ (\hat{x}, w) \in R_{\hat{L}} \vee \hat{y}_b = f(w)\}$. Note that for any string $\hat{x}$ (whether $\hat{x} \in \hat{L}$ or not), the statement "$(\hat{x}, y_0, y_1) \in L$" is always true as $PK = (y_0, y_1)$ is honestly generated; that is, the language L is dependent upon the OWF f used by the honest verifier in key generation, and the verifier's registered public key is instantiated as part of a (true) statement in L. Also note that L is a language that admits Σ protocols (as Σ_{OR} protocol itself is a Σ protocol).

Now, we describe the concurrent interleaving and malleating attack, in which P^* successfully convinces the honest verifier of the statement "$(\hat{x}, y_0, y_1) \in L$" for *any arbitrary* string $\hat{x}$ (*even when* $\hat{x} \notin \hat{L}$) by concurrently interacting with V in two sessions. For ease of understanding, the message schedule of P^* in the two sessions is also diagrammatically presented in Fig. 3.

1. P^* initiates the first session with V and works just as the honest prover does in Phase 1 and Phase 2 of the first session.
2. Denote by $c_{\hat{e}}$ the Phase 2 message of the first session, i.e., $c_{\hat{e}}$ commits to a random string $\hat{e}$ of length n.
3. When P^* moves into Phase 3 of the first session and needs to send V the first-round message, denoted a_P, of the Σ protocol of Phase 3 of the first session *on common input* $(\hat{x}, y_0, y_1)$, P^* suspends the first session and does the following:
   - P^* first runs the SHVZK simulator of $\Sigma_{\hat{L}}$ (i.e., the Σ protocol for $\hat{L}$) on $\hat{x}$ to get a simulated conversation, denoted $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$."

- Then, P^* initiates the second session with V (For OWF-based implementation, P just sends $r_P = r_V$ as its first message to V, where r_V is the random string registered by V as a part of its public key for OWF-based implementation.). After receiving the first-round message, denoted a'_V, of the Σ_{OR} protocol of Phase 1 of the second session on common input (y_0, y_1) (i.e., V's public key), P^* sets $a_P = (a_{\hat{x}}, a'_V)$ and suspends the second session.

4. Now, P^* continues the execution of the first session and sends $a_P = (a_{\hat{x}}, a'_V)$ to V as the first-round message of the Σ protocol of Phase 3 of the first session, where a'_V is the one received by P^* in the second session.
5. After receiving the second-round message of Phase 3 of the first session, denoted q (i.e., the random challenge from V), P^* sets $e_P = \hat{e} \oplus q$, where $\hat{e}$ is the random string committed to $c_{\hat{e}}$ at Phase 2 of the first session, and then suspends the first session again.
6. P^* continues the second session and sends $e'_V = e_P \oplus e_{\hat{x}} (= \hat{e} \oplus q \oplus e_{\hat{x}})$ as the second-round message of the Σ_{OR} protocol of Phase 1 of the second session.
7. After receiving the third-round message of the Σ_{OR} protocol of Phase 1 of the second session, denoted z'_V, P^* suspends the second session again.
8. P^* continues the execution of the first session again, reveals $\hat{e}$ committed to $c_{\hat{e}}$ at Phase 2 of the first session and sends to V $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V))$ and the decommitment information of $\hat{e}$ in the last-round message of the first session.

Note that $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$ is an accepting conversation for the (possibly false) statement "$\hat{x} \in \hat{L}$," and (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_0 or y_1, and furthermore, $e_{\hat{x}} \oplus e'_V = e_P = \hat{e} \oplus q$. According to the description of Σ_{OR} (cf. Sect. 2), 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 $(\hat{x}, y_0, y_1)$. That is, P^* successfully convinced V of the statement "$(\hat{x}, (y_0, y_1)) \in L$" (*even if* $\hat{x} \notin \hat{L}$ or $\hat{x}$ is just an empty string) in the first session *but without knowing any corresponding* $\mathcal{NP}$*-witness*. This demonstrates that the protocol of [28,82] also fails to be a proof of knowledge (fails knowledge extraction) in concurrent executions (note that it was not designed as such, since this new issue is the notion we put forth here). We note that the above attack does not violate the concurrent soundness of the CZK protocol of [28], which does indeed hold even though the malicious prover can set languages and statements depending upon verifiers' public keys. Specifically, with our attack, the statement $(\hat{x}, y_0, y_1)$ is always a true statement for the public-key-dependent language L. But our attack shows that, when the underlying language and statements may be dependent upon verifiers' public keys, the CKE property of the protocol of [28] loses in general (particularly considering $\hat{x} \notin \hat{L}$ or $\hat{x}$ is just an empty string), which indicates a gap between concurrent soundness and concurrent knowledge extraction in the public-key model (see more clarifications in Proposition 5.1, p. 30).

Remark. The above attack is demonstrated w.r.t. a concrete language with the Σ_{OR} technique, where the dependency of the language L and statements (being proved) upon verifier's public key PK can be trivially checked. In general, the language L can be any $\mathcal{NP}$-complete language, and a concurrent malicious prover P^* can adaptively select (public-key-dependent) statements for all sessions and then reduce the statements by

$\mathcal{NP}$-reductions to the statements for the $\mathcal{NP}$-complete language L. In this case, in general it is hard to efficiently determine whether a statement generated by P^* for the $\mathcal{NP}$-complete language L is related to verifier's public key or not. We suggest that mixing the public-key structure as part of the language and statements is a natural adversarial strategy against cryptographic systems in the public-key model.

5. Formulating Concurrent Knowledge Extraction in the Public-Key Model

Now, we proceed to formulate concurrent verifier security in light of the above concrete attacks presented in Sect. 4. Note that these concrete attacks are of *man-in-the-middle* (MIM) nature and are related to malleability of protocols. The security notion assuring that a malicious prover P^* does "know" what it claims to know, when it is concurrently interacting with the honest verifier V in the public-key model, can informally be described as follows. For any x, if P^* can convince V (with public key PK) of "$x \in L$" (for an $\mathcal{NP}$-language L) by concurrent interactions, then there exists a PPT knowledge extractor that outputs a witness for $x \in L$. This is a natural extension of the normal arguments of knowledge into the concurrent settings in the public-key model. However, such a definition does *not* work for *polynomial-time black-box* knowledge extraction in the public-key model, as is the focus of this work. The reason is: The statements being proved may be related to PK, and thus, the extracted witness may be related to its corresponding secret key SK.[8] However, in knowledge extraction, the PPT extractor may have already possessed SK (the difficulty or impossibility of CKE directly from PK without knowing SK is discussed and clarified in Sect. 5.1). To solve this subtlety, we require the extracted witness, together with adversary's view, to be *independent* of SK. However, the problem here is how to formalize such independence, in particular, w.r.t. a concurrent MIM? We solve this in the spirit of non-malleability formulation [31]. That is, we consider the message space (distribution) of SK, and such independence is roughly formulated as follows: letting SK be the secret key and SK' be an element randomly and independently distributed over the space of SK, then we require that, for any polynomial-time computable relation R, the probability $\Pr[R(\bar{w}, SK, view) = 1]$ be negligibly close to $\Pr[R(\bar{w}, SK', view) = 1]$, where $\bar{w}$ is the set of witnesses extracted by the knowledge extractor for successful concurrent sessions and $view$ is the view of the adversary P^*. This captures the intuition that P^* does, in fact, "know" the witnesses to the statements whose validations are successfully conveyed by concurrent interactions.

Definition 5.1. *(concurrent knowledge extraction (CKE) in the public-key model)* We say that a protocol $\langle P, V\rangle$ is concurrently knowledge extractable in the BPK model w.r.t. some key-validating relation R_{KEY} and some $\mathcal{NP}$-relation $\mathcal{R}_L$, if for any positive polynomial $s(\cdot)$, any s-concurrent malicious prover P^* defined in Sect. 2, there exists a pair of (expected) polynomial-time algorithms S (the simulator) and E (the extractor) such that for any sufficiently large n, any auxiliary input $z \in \{0, 1\}^*$ and any polynomial-time computable relation R (with components drawn from $\{0, 1\}^* \cup \{\perp\}$), the followings hold, in accordance with the experiment $\mathbf{Expt_{CKE}}(1^n, z)$ described below (p. 26):

[8] Actually, for the malicious prover strategies of the concrete attacks presented in Sect. 4, the extracted witness will just be the same secret key used by the knowledge extractor.

Expt$_{\mathbf{CKE}}(1^n, z)$

The simulator $S = (S_{KEY}, S_{PROOF})$**:**
$(PK, SK, SK') \longleftarrow S_{KEY}(1^n)$, where the distribution of (PK, SK) is identical with that of the output of the key-generation stage of the honest verifier V, $R_{KEY}(PK, SK) = R_{KEY}(PK, SK') = 1$ and the distributions of SK and SK' are identical and *independent*. In other words, SK and SK' are two random and independent secret keys corresponding to PK.
$(str, sta) \longleftarrow S_{PROOF}^{P^*(1^n, PK, z)}(1^n, PK, SK, z)$; that is, on input $(1^n, PK, SK, z)$ and with oracle access to $P^*(1^n, PK, z)$, the simulator S outputs a simulated transcript str, and some state information sta to be transferred to the knowledge extractor E.
We denote by $S_1(1^n, z)$ the random variable str (in accordance with the above processes of S_{KEY} and S_{PROOF}). For any $(PK, SK) \in R_{KEY}$ and any $z \in \{0, 1\}^*$, we denote by $S_1(1^n, PK, SK, z)$ the random variable describing the first output of $S_{PROOF}^{P^*(1^n, PK, z)}(1^n, PK, SK, z)$, i.e., str specific to (PK, SK).
The knowledge extractor E**:**
$\overline{w} \longleftarrow E(1^n, sta, str)$. On input (sta, str), E outputs a list of witnesses to statements whose validations are successfully conveyed in str.

- **Simulatability.** The following two ensembles are identical (or computationally indistinguishable in general).[9]

$$\left\{S_1(1^n, PK, SK, z)\right\}_{(PK,SK)\in R_{KEY}, z\in\{0,1\}^*}, \text{ and}$$
$$\left\{view_{P^*}^{V(SK)}(1^n, z, PK)\right\}_{(PK,SK)\in R_{KEY}, z\in\{0,1\}^*} \text{ (cf. Section 3.2).}$$

- **Secret key independent knowledge extraction.** E, on input $(1^n, str, sta)$, outputs witnesses to all statements successfully proved in accepting sessions in str. Specifically, E outputs a list of strings $\overline{w} = (w_1, w_2, \ldots, w_{s(n)})$, satisfying the following:
  - w_i is set to be $\perp$, if the ith session in str is not accepting (due to abortion or verifier verification failure), where $1 \le i \le s(n)$.
  - Correct knowledge extraction for (individual) statements: In all other cases (i.e., for successful sessions), it holds that $(x_i, w_i) \in R_L$ with overwhelming probability, where x_i is the common input selected by P^* for the ith session in str.
  - (Joint) knowledge extraction independence (KEI): $\Pr[R(SK, \overline{w}, str) = 1]$ is negligibly close to $\Pr[R(SK', \overline{w}, str) = 1]$.

  The probabilities are taken over the randomness of S in the key-generation stage (i.e., the randomness for generating (PK, SK, SK')) and in all proof stages, the randomness of E, and the randomness of P^*. If the KEI property holds for any (not necessarily polynomial-time computable) relation R, we say the protocol $\langle P, V\rangle$ satisfies *statistical* CKE and *statistical* KEI.

[9] For all CZK–CKE protocols achieved in this work, we actually have perfect simulatability.

5.1. *Discussion and Justification of the CKE Formulation*

We first note that the above CKE formulation follows the simulation-extraction approach of [70] (which is also used in [7]). Here, the key augmentation, besides some other adaptations in the public-key model, is that the property of knowledge extraction independence (KEI) is explicitly required. Though the CKE and KEI notions are formulated in the public-key model, they are actually applicable to protocols in the plain model, in general, in order to capture knowledge extractability against concurrent adversaries interacting with honest players of secret values.

On extending the Bellare-Goldreich (BG) quantitative approach for stand-alone POK into the concurrent setting. We note that, besides the subtle KEI issue, there are some difficulties (or inconveniences) to extend the Bellare-Goldreich quantitative approach for stand-alone POK [8,9,38] (i.e., the quantitative definition of expected knowledge extraction time that is in inverse proportion to the probability the adversary convinces of the statement) into the concurrent setting. Below, we consider two possible approaches to extend the Bellare-Goldreich quantitative approach (for stand-alone POK) into the concurrent setting.

The first approach (as considered in [30]) is: For each of the concurrent sessions, we consider the probability that the adversary (i.e., the concurrent malicious prover P^*) successfully finishes the session. Denote by p_i the probability that the adversary successfully finishes the ith session. Note that this probability is particularly taken over the random coins of P^* and all random coins of the honest verifier instances in all concurrent sessions. However, within the simulation-extraction formulation framework, it is difficult to give a precise quantitative definition of the knowledge extraction time inversely proportional to p_i. The reason is as follows when we apply the underlying stand-alone knowledge extractor (guaranteed by the Bellare-Goldreich POK definition) on the successful ith session in the simulated transcript, the knowledge extraction is actually with respect to the probability, denoted p'_i, that P^* successfully finish the ith session when the coins of the honest verifier instances in all other sessions except the ith one are fixed (i.e., determined by the simulated transcript). Clearly, p'_i can be totally different from p_i (e.g., p_i may be non-negligible, but p'_i can be negligible), and thus, the knowledge extraction time w.r.t p'_i can be totally different from that w.r.t p_i in general.

The second approach is to separate the simulation and knowledge extraction. Specifically, besides indistinguishable simulation, we *separately* require (regardless of the simulated transcript) that for any statement x *selected adaptively by the adversary on the fly in the concurrent attack*, if the adversary P^* can, with probability p_x, convince the honest verifier of the statement "$x \in L$" (in one of the $s(n)$ sessions) by concurrent interactions, the knowledge extraction time should be in inverse proportion to p_x. We note that this approach does not work. On the one hand, suppose P^* convinces $x \in L$ in one of the $s(n)$ sessions (e.g., the ith session) with some non-negligible probability, but with negligible probability in all other sessions. In this case, it is okay if the knowledge extraction is w.r.t the ith session, but will fail w.r.t other sessions. On the other hand, one may argue that to remedy the above subtlety, we can add a (polynomial-time) bound on the knowledge extraction in each session, but this solution fails if the adversary convinces of the statement "$x \in L$" with negligible probability in all sessions. In general, it may be hard to predict the two different cases for knowledge extraction, i.e., the case when P^* succeeds with negligible probability in all sessions and the case when P^* may succeed with non-negligible probability in some (but not all) sessions.

CKE with secret keys versus CKE with public keys In our CKE formulation, the simulator/extractor possesses verifier's secret key w.r.t. a simulated public key. In this case, explicitly requiring the KEI property is crucial for correctly formulating CKE, or otherwise in general, we cannot ensure that the concurrent malicious prover P^* does indeed know the witnesses to statements being successfully proved. A natural and intuitive strengthening of the CKE formulation might be the simulator/extractor only uses the public key of the honest verifier. Specifically, for any concurrent malicious P^* there exists a PPT simulator/extractor that, *only on the public key of the honest verifier (without knowing the corresponding secret key)*, outputs an indistinguishable simulated transcript together with all the witnesses to accepting sessions. In this case, as the simulator/extractor does not possess the secret key of the honest verifier, the KEI property can be waived. Needless to say, CKE only with verifier public key is more desirable, which also better fits the spirit of non-malleability. By comparison, CKE with verifier's secret key mainly ensures that, whenever an honest verifier is convinced of a list of $\mathcal{NP}$-statements (chosen adaptively by P^* via concurrent interactions), the corresponding witnesses can be efficiently extracted by the honest verifier (of public key PK and secret key SK) itself, and the extracted witnesses are independent of verifier's secret key.

Below, we discuss some difficulties of achieving CZK protocols with *black-box polynomial-time* CKE in the BPK model, as is the focus of this work. The first observation is: We do not know how to achieve, with known construction techniques, constant-round *black-box polynomial-time* CKE (*whether ZK or not*) with only public keys. In particular, with known techniques for constant-round black-box CZK in the BPK model, constant-round CZK–CKE with only public keys will imply constant-round CZK (actually, possibly concurrently non-malleable ZKAOK) *in the plain model* by viewing verifiers' public keys as a part of common inputs, which is, however, impossible at least in the black-box sense [16]. Roughly speaking, the underlying reason is that, in order for a protocol in the BPK model to become CKE secure *with only public keys*, all messages generated by the honest verifier using its secret key should be efficiently simulatable (i.e., zero knowledge) according to the CKE definition. In more detail, a constant-round black-box CZK protocol in the BPK model is usually composed of two sub-protocols: One sub-protocol is referred to as the verifier sub-protocol, in which V plays the role of the prover using its secret key as the witness w.r.t. a statement relative to V's public keys;[10] and the other sub-protocol is referred to as the prover sub-protocol, in which the prover P proves to V on the common statement and V's public keys. Note that we assume provers do not register keys in the BPK model. Then, by the *simulatability* property required in the CKE formulation, CKE with only public keys implies that the verifier sub-protocol is CZK (i.e., the concurrent interactions of the verifier sub-protocol can be

[10] To our knowledge, some exceptions are the constant-round concurrent/resettable ZK protocols in the public-key model [27] or in the bounded player model [51], where verifiers' secret keys are used only for generating signatures. This paradigm, i.e., using signatures for achieving round-efficient concurrent/resettable ZK, was introduced in [27] and was also used in [18,60] for other reasons. But as signatures are not efficiently simulatable, the round-efficient concurrent/resettable ZK protocols with this approach cannot be CKE secure *merely with public keys*. Indeed, the CKE issue is beyond the consideration there.

simulated in polynomial time) *in the plain model*. Moreover, a malicious prover actually plays the CMIM role between the verifier sub-protocol and the prover sub-protocol. If some building tools are employed in both sub-protocols, to ensure concurrent knowledge extraction we actually need the verifier CZK sub-protocol to be secure against CMIM attacks.

On the other hand, as constant-round CZKAOK exists in the BPK model [28,68, 75,77], one may propose to additionally require the honest provers also to register public keys, and then let verifier use a constant-round CZKAOK protocol to prove the knowledge of its secret key in the verifier sub-protocol. For ease of reference, we refer to the BPK model variant, where both provers and verifiers register public keys, as *generalized* BPK model. The problems with this approach are as follows.

- Firstly, consider potential CZK–CKE solutions with this approach, i.e., using CZKAOK in the generalized BPK model as the verifier sub-protocol. For all the existing CZKAOK protocols [28,68,75,77] (to be used as the verifier sub-protocol in the generalized BPK model), the CZK simulator S for the whole composed protocol, say the potential CZK–CKE solution, needs to possess the secret keys of honest provers. But CZK simulation (for the whole composed protocol) in the generalized BPK model should be based only upon honest provers' public keys. That is, the actions of the prover instances in the generalized BPK model should be efficiently simulated from the common statements and their public keys (without knowing the corresponding $\mathcal{NP}$-witnesses or their secret keys).
- Secondly, even if we allow the CZK simulator S to possess honest provers' secret keys, we need to ensure that: (1) the knowledge extracted by the CZK simulator S from the malicious verifier V^* should just be the secret keys corresponding to public keys registered by V^*; and (2) the knowledge extracted should not be efficiently derived from honest provers' secret keys possessed by S, or otherwise, in general, V^* may not necessarily “know" the knowledge extracted or the CZK simulation (with honest provers' secret keys) can be trivially meaningless. Note that V^* may register public keys based on honest provers' public keys. In a sense, we need to require the underlying constant-round CZKAOK protocols in the BPK model (used as the verifier sub-protocol) have already been CKE secure, while CKE is just our goal to achieve.

Another approach to bypass the black-box CKE obstacle is to employ a constant-round *non-black-box* concurrent (non-malleable) ZK protocol by verifiers to prove the knowledge of their secret keys. In this case, knowledge extraction is also non-black-box, i.e., the knowledge extractor needs to possess the code of the malicious prover P^*. However, achieving constant-round non-black-box concurrent ZK protocols *under standard (black-box) complexity assumptions* is a fundamental open problem in the literature. Also, black-box CKE and non-black-box CKE are incomparable in general, and as discussed in [3,4], black-box simulation and knowledge extraction is more desirable than their non-black-box counterparts. Finally, we mention that CZK with black-box *sub-exponential-time* CKE (only with verifiers' public keys) in the BPK model is still possible, which is implied in [80] but is meaningful only for sub-exponentially hard languages.

We also note that knowledge extraction with verifier secret key is not unique to this work. To our knowledge, for most existing *black-box* constant-round CZK (stand-alone or sequential) AOK protocols in the BPK model (e.g., [28,68,75,77]), when arguing about the black-box stand-alone or sequential AOK property, it is implicitly assumed that the knowledge extractor possesses verifier's secret key w.r.t. a simulated public key. One exception is the work [4], which particularly achieves constant-round (non-round-optimal) CZK with *non-black-box* sequential AOK in the BPK model, where the knowledge extractor does not possess verifier's secret key but possesses the code of the concurrent malicious prover.

Taking adversary's view into account for capturing KEI. We note that explicitly taking account of the view of the adversary, i.e., str, is necessary for the completeness of KEI formulation. Specifically, consider the following (seemingly impossible) case that: for any extracted $w_i \in \bar{w}$, $w_i = PRF_s(SK)$, where the seed s could be either a part of the adversary's random tape or a value computed from its view. In other words, the witnesses extracted are always dependent on the secret key used by the simulator/extractor, and thus, the adversary may not necessarily be aware of the extracted knowledge. Clearly, such cases are excluded by taking account of adversary's view in the KEI formulation. Note that the KEI definition is quantified over any polynomial-time relations. However, without taking account of adversary's view, $\Pr[R(SK, \bar{w}) = 1]$ is still negligibly close to $Pr[R(SK', \bar{w}) = 1]$ in this case for any polynomial-time computable relation R.

More on the KEI Property. With the KEI definition, we are actually formulating the independence of the witnesses, used ("*known*") by the concurrent malicious prover P^*, on the secret key (witness) used by the verifier who may in turn play the role of knowledge prover in some sub-protocols. Note that a malicious prover P^* can adaptively select the language and statements to be proved in the concurrent sessions, and mixing the public-key structure as part of the languages and statements can be a natural attack strategy against cryptographic systems in the public-key model.

We consider some special cases where the KEI property may be trivially satisfied or waived. Firstly, in case the languages and statements being proved are independent of verifiers' public keys, it seems that we can trivially get rid of the KEI property. However, there are some problems with this observation.

- If a CKE protocol is used as a building tool or sub-protocol in a larger system in the public-key model, as users' keys play an essential role (e.g., for round efficiency) in cryptographic systems in the public-key model, the actual languages and statements to the underlying CKE protocol (within the run of the larger system) are usually related to users public keys. In other words, restricting CKE protocols to work only for languages and statements independent of users' keys can significantly limit the applicability of CKE protocols.
- In general, it is hard to efficiently detect the dependency between verifier's public key and statements being proved. For example, supposing the protocol is for an $\mathcal{NP}$-complete language, P^* can form arbitrary languages and statements (some of them may be related to verifier's public key PK, and some of them may be not), and then reduce the arbitrary languages and statements to the underlying $\mathcal{NP}$-complete language. In this case, in general it seems hard to efficiently determine

the dependency between PK and the statements chosen by P^*, even if all the secret keys corresponding to PK are known.

Consider another case where verifier's secret key has a special (unnatural) structure such that knowing a secret key allows one to efficiently compute any other secret key. In this case, we can have an extractor that outputs a random and independent secret key. In case any secret key of the verifier is always a witness to all the statements being proved by the malicious prover P^*, the CKE property can be trivially satisfied. However, the problem with this observation is as follows.

- To our knowledge, all known *natural* OWFs underlying key generation do not satisfy this property, i.e., knowing one secret key allows efficiently computing any other secret key.
- In general, the random and independent secret key, computed and output by the extractor based on the knowledge of the secret key possessed, is not necessarily, or can be used to derive, a witness to the statements being proved by the malicious prover P^*. For example, the protocol is designed for a language that is independent of PK, or for an $\mathcal{NP}$-complete language. Recall that the CKE formulation requires correct knowledge extraction for all the statements being successfully proved.

We also note that our KEI formulation implicitly assumes that verifier's public key corresponds to multiple secret keys, which, however, can typically be achieved with the common key-pair trick [67]. In general, we suggest cryptography literature may welcome diversified approaches for modeling and achieving security goals of cryptographic systems, particularly witnessed by the history of public-key encryption.

CKE versus concurrent soundness We show that, assuming any OWF, CKE is a *strictly* stronger notion for concurrent verifier security than concurrent soundness in the public-key model.

Proposition 5.1. *Assuming any OWF, CKE is strictly stronger than concurrent soundness in the public-key model. In particular, there exist protocols in the BPK model that are concurrently sound CZK arguments of knowledge but not concurrent knowledge extractable.*

Proof. It is easy to see that CKE implies concurrent soundness in the public-key model. Specifically, supposing that for some $(PK, SK) \in R_{KEY}$, some $\mathcal{NP}$-language L and some string $x \notin L$, P^* can convince $V(SK)$ of the false statement "$x \in L$" with non-negligible probability in real execution, then with essentially the same probability (up to at most a negligible gap) P^* can convince the simulator $S(SK)$ of $x \in L$ in $\mathsf{Expt}_{\mathrm{CKE}}(1^n, z)$ by the property of simulatability, which, however, contradicts the secret key independent knowledge extraction property.

Then, the proposition is direct from the attack demonstrated in Sect. 4.2.2 on the CZK protocol of [28] that is both concurrently sound and normal argument of knowledge and can be implemented based on any OWF. Specifically, for the specific strategy of P^* of the concurrent interleaving and malleating attack presented in Sect. 4.2.2, supposing $\hat{x} \notin \hat{L}$ or just $\hat{L}$ is empty, the witness extracted by any polynomial-time knowledge extraction algorithm E (with $SK = s_b$ as its input) must be the preimage of either y_0 or y_1. However, according to the one-wayness of f used in the key-generation stage, with overwhelming probability the extracted witness will be the preimage of y_b conditioned on E outputs a witness. Specifically, consider the simulator/extractor emulates the key generation of the honest verifier, except that the value y_{1-b} is received externally as its input. Now, define the relation R to be $R(w, SK, \cdot) = 1$ if $f(w) = f(SK)$. Then, conditioned on E outputs a witness, with overwhelming probability the extracted witness (i.e., the preimage of y_b) is always related to $SK = s_b$, but can be related to a random and independent SK' with negligible probability under the relation R. Thus, the CZK protocol of [28] is not concurrently knowledge extractable in the public-key model. $\square$

6. Generic CZK–CKE in the BPK Model

In this section, we present the generic constant-round CZK–CKE arguments for $\mathcal{NP}$ in the BPK model under standard hardness assumptions. The starting point is the basic FSZK structure [35,36]. For the FSZK structure presented in Sect. 4.1, letting (y_0, y_1) serve as the public key of V and s_b (for a random bit b) as the secret key, it is shown in [82] that such a public-key version of FSZK is CZK in the BPK model. However, as we just demonstrated in Sect. 4.1, the public-key version of FSZK is neither concurrently knowledge extractable nor concurrently sound in the BPK model (indeed, FSZK was not designed for the public-key model). We hope to add the CKE property to FSZK in the BPK model (and thus get concurrent security both for the prover and for the verifier simultaneously), while retaining its conceptually simple structure as well as the ability of practical instantiations.

The subtle point here is: We are actually facing and dealing with a concurrent MIM adversary, who manages to malleate, in a malicious and unpredictable way, the public keys and knowledge proof interactions of the verifier in one session into the statements and knowledge proof interactions in another concurrent session. To add CKE security to FSZK in the BPK model, some non-malleable (may be inefficient) building tools seem to be intrinsically required. In this work, we show how to do so without employing any non-malleable building tool.

The idea is to strengthen the first sub-protocol to be *statistical* WIPOK and require the prover to first commit, before starting the second WI sub-protocol, the supposed witness to c_w by running a *statistically binding* commitment scheme C. This guarantees that if the witness committed to c_w is dependent on the secret key used by V, there are indeed some differences between the interaction distribution when V uses $SK = s_0$ and that when V uses $SK = s_1$, and we can then use such distribution differences to violate the statistical WI of the first sub-protocol. However, this solution loses CZK in general, as the second WI sub-protocol is run w.r.t. commitments to different values in real interactions and in the simulation. This problem can be overcome by using a stronger second sub-protocol, i.e., the strong WI (SWI) [38]. We show that the composition of commitment and SWI itself is regular WI, and thus, CZK property is retained.

The generic construction is depicted in Fig. 4 (p. 32). As we shall see in Sect. 7.1, we can also have a practical (8-round) instantiation of this generic CZK–CKE construction under the DDH assumption for the DLP-based number-theoretic language without going through general $\mathcal{NP}$-reductions.

Key Generation. Let $f : \{0,1\}^n \rightarrow \{0,1\}^m$ be any OWF, where 1^n is the system security parameter. Each verifier V randomly and independently selects two strings s_0, s_1 from $\{0,1\}^n$, randomly selects a bit $b \leftarrow \{0,1\}$, computes $y_b = f(s_b)$ and sets $y_{1-b} = f(s_{1-b})$. V registers $PK = (y_0, y_1)$ in a public file F as its public key, and keeps $SK = s_b$ as its secret key. Define $R_{KEY} = \{((y_0, y_1), s) | y_0 = f(s) \vee y_1 = f(s)\}$.

Common input. An element $x \in L \cap \{0,1\}^{poly(n)}$, where L is any $\mathcal{NP}$-complete language with the corresponding $\mathcal{NP}$-relation R_L.

P's private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^{poly(n)}$ for $x \in L$. Here, we assume without loss of generality that the witness for any $x \in L \cap \{0,1\}^{poly(n)}$ is of the same length $poly(n)$.

Stage-1. V proves to P that it knows the preimage of y_0 or y_1, by running a *statistical* WIA/POK protocol for $\mathcal{NP}$, in which V plays the role of knowledge prover. The witness used by V in this stage is s_b.

Stage-2. If V successfully finishes Stage-1, P does the following: it computes and sends $c_w = C(w, r_w)$, where C is a statistically-binding commitment scheme and r_w is the randomness used for committing to w.

Stage-3. Define a new $\mathcal{NP}$-language $L' = \{(x, y_0, y_1, c_w) | \exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge [(x, w) \in R_L \vee (y_0 = f(w) \vee y_1 = f(w))]\}$ Then, P proves to V that it knows a witness for $(x, y_0, y_1, c_w) \in L'$, by running a strong WI argument/proof of knowledge (WIA/POK) protocol for $\mathcal{NP}$.

Fig. 4. Generic CZK–CKE argument $\langle P, V \rangle$ for $\mathcal{NP}$ in the BPK model.

6.1. *Security Analysis*

Notes on the underlying hardness assumptions and round complexity. If the OWF f used in key generation admits statistical Σ protocols (and thus, we can use Σ_{OR} in Stage 1), and we use Feige–Shamir ZK (FSZK) of [36] (with WI is replaced by Σ_{OR}) to replace SWI of Stage 3, the protocol depicted in Fig. 4 can be based on any OWF admitting statistical Σ protocols and runs in eight rounds in total. Recall that any (auxiliary input) zero-knowledge protocol itself is strong WI (cf. Proposition 4.6.3 of [38]). If we use in Stage 1 the modified Blum's protocol for DHC with constant-round statistically/perfectly hiding commitments (cf. Section 2, p. 14) and FSZK in Stage 3, the protocol depicted in Fig. 4 can be based either on any claw-free collection with efficiently recognizable index set or on any collision-resistant hash function and runs in nine rounds. Here, if one party uses Naor's OWF-based 2-round statistically binding commitment scheme C, the first-round message of C can be merged with the last preceding message from the other party.

Commit-then-SWI is regular WI. Consider the following protocol composing a statistically binding commitment and SWI:

Common input: $x \in L$ for an $\mathcal{NP}$-language L with corresponding $\mathcal{NP}$-relation R_L.
Prover auxiliary input: w such that $(x, w) \in R_L$.
The commit-then-SWI protocol (consisting of two stages):

Stage- 1: The prover P computes and sends $c_w = C(w, r_w)$, where C is a statistically binding commitment and r_w is the randomness used for commitment.

Stage- 2: Define a new language $L' = \{(x, c_w) | \exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge R_L(x, w) = 1\}$. Then, P proves to V that it knows a witness to $(x, c_w) \in L'$, by running a SWI protocol.

One interesting observation for the above commit-then-SWI protocol is that commit-then-SWI itself is a regular WI for L.

Proposition 6.1. *Commit-then-SWI itself is a regular WI for the language L.*

Proof. (of Proposition 6.1). For any PPT malicious verifier V^* possessing some auxiliary input $z \in \{0,1\}^*$, and for any $x \in L$ and two (possibly different) witnesses (w_0, w_1) such that $(x, w_b) \in R_L$ for both $b \in \{0,1\}$, consider the executions of commit-then-SWI: $\langle P(w_0), V^*(z)\rangle(x)$ and $\langle P(w_1), V^*(z)\rangle(x)$.

Note that for $\langle P(w_b), V^*(z)\rangle(x)$, $b \in \{0,1\}$, the input to SWI of Stage 2 is $(x, c_{w_b} = C(w_b, r_{w_b}))$, and the auxiliary input to V^* at the beginning of Stage 2 is (x, c_{w_b}, z). Note that (x, c_{w_0}, z) is indistinguishable from (x, c_{w_1}, z). Then, the regular WI property of the whole composed protocol follows from the SWI property of Stage 2. □

Theorem 6.1. *The protocol depicted in Fig. 4 is a constant-round concurrently knowledge extractable concurrent zero-knowledge (CZK–CKE) argument for $\mathcal{NP}$ in the BPK model.*

Proof. The completeness of the protocol $\langle P, V\rangle$ can be easily checked.

Concurrent zero knowledge.

To build a black-box expected polynomial-time CZK simulator S from scratch, where S does not know at the onset any secret keys corresponding to public keys in the public file, we resort to the techniques proposed in [15,75].

For any $s(n)$-concurrent malicious verifier V^* (defined in Sect. 3) and any $\mathcal{NP}$-language L, the simulator S runs V^* as a subroutine on input $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$ (where x_k might equal $x_{k'}$, $1 \le k \ne k' \le s(n)$), and the public file $F = (PK_1, \ldots, PK_{s(n)})$. The simulator maintains a *straight-line main-thread* of the simulation, while invoking at most $s(n)$ *extraction threads*. Specifically, whenever the malicious verifier V^* (run by S) successfully finishes Stage 1 of the ith session w.r.t. an "uncovered" public key PK_j (for which S does not know the corresponding secret key SK_j yet) in the main-thread simulation, $1 \le i \le s(n)^2$ and $1 \le j \le s(n)$, S suspends the main-thread simulation and invokes an *extraction thread*, denoted $Ext(i, j)$, to extract (i.e., "cover") the secret key SK_j by running the knowledge extractor ensured by the WIA/POK of Stage 1. After SK_j is covered, S resumes the main-thread simulation by using SK_j as the witness in generating Stage 2 and Stage 3 messages for all sessions w.r.t. PK_j. In order to get a modular analysis, we actually apply Lindell's witness-extended emulator as per Definition 2.8 in the extraction thread. The simulator S is described in Fig. 5 (p. 34).

For any i, j, $1 \le i \le s(n)^2$ and $1 \le j \le s(n)$, and for any $\mathcal{C}_{i,j}$ and any $\omega_{i,j}$, denote by $p_{i,j}$ the probability that the extraction thread $Ext(i, j)$ is invoked (w.r.t. $\omega_{i,j}$ and $\mathcal{C}_{i,j}$) in the main-thread simulation. Denote by $p'_{i,j}$ the probability that the combined determined knowledge prover $\hat{P}^{(i,j)}$ (defined for $Ext(i, j)$) convinces the stand-alone knowledge verifier of the statement "$PK_j \in L_{KEY}$" (in other words, the probability that E_1 outputs a successful conversation). By the specification of $\hat{P}^{(i,j)}$ defined for $Ext(i, j)$, we have that $p_{i,j} = p'_{i,j}$. Then, by the witness-extended emulation lemma [61] (cf. Lemma 2.1, Section 2), we have that each extraction thread takes expected polynomial time

Security parameter: 1^n.
Common input: $\bar{\mathbf{x}} = (x_1, \cdots, x_{s(n)}) \in L^{s(n)}$.
Main-thread transcript: tr, which is initialized to be $\{\bar{x}\}$.
Covered-key set: $\mathcal{C}$, which is initialized to be an empty set.

Main-thread simulation: During the simulation, the simulator S always keeps updating and extending tr to include the transcript generated in the main-thread so far, which is not always explicitly specified for presentation simplicity.

Step-1: Select a random tape for V^*, and run the key-generation phase of V^* to obtain the public file $F = (PK_1, \cdots, PK_{s(n)})$, and extend tr to include F.

Step-2: Run $V^*(tr)$ (i.e., run V^* starting from the updated transcript tr), and work as follows (to further extend tr):

Case-1. V^* stops. In this case, S returns the main-thread transcript tr generated so far and stops.

Case-2. V^* fails in successfully finishing some session. In this case, S aborts that session, and goes to **Step-2** (with extended transcript).

Case-3. V^* successfully finishes the statistical WIA/POK of Stage-1 in some session w.r.t. a *covered* public key $PK_j \in \mathcal{C}$, $1 \le j \le s(n)$. In this case, S generates Stage-2 and Stage-3 messages of that session (and any subsequent sessions w.r.t. PK_j) by using the extracted secret key SK_j as the witness.

Case-4. V^* successfully finishes the statistical WIA/POK of Stage-1 in the i-th session, $1 \le i \le s(n)^2$, w.r.t. an *uncovered* public key $PK_j \notin \mathcal{C}$, $1 \le j \le s(n)$. In this case, S suspends extending the main-thread transcript tr, invokes the extraction-thread procedure $Ext(i,j)$ described below (in order to extract the secret key SK_j with overwhelming probability in expected polynomial-time). If the output returned by $Ext(i,j)$ is "$\perp$", S aborts and outputs "$\perp$" indicating simulation failure. If $Ext(i,j)$ returns back SK_j, S sets $\mathcal{C} = \mathcal{C} \bigcup \{(PK_j, SK_j)\}$, and goes to **Step-2** to resume extending the main-thread tr, where the Stage-2 and Stage-3 messages of the i-th session (and any subsequent sessions w.r.t. PK_j) will be generated by using SK_j as the witness.

Extraction-thread $Ext(i,j)$**:**

1. Denote by $\mathcal{C}_{i,j}$ the updated covered-key set upon invoking $Ext(i,j)$, and by $tr_{i,j}$ the extended main-thread transcript just before V^* sends the first-round message of Stage-1 of the i-th session w.r.t. a public key PK_j in the main-thread simulation. Denote by $\omega_{i,j}$ the set of random coins for V^* and the random coins used by the simulator S in the main-thread simulation except those to be used for simulating the prover messages in Stage-1 of the i-th session.

   Combine V^* and the main-thread simulation of the simulator S excluding the random coins used by S for simulating Stage-1 of the i-th session, including $tr_{i,j}$, $\mathcal{C}_{i,j}$, $\omega_{i,j}$, into a deterministic knowledge prover $\hat{P}^{(i,j)}$. The knowledge-prover $\hat{P}^{(i,j)}$ only interacts with a stand-alone knowledge-verifier of the underlying statistical WIA/POK of Stage-1, by running $V^*(tr_{i,j})$ (i.e., running V^* starting from $tr_{i,j}$) internally and mimicking S in the main-thread simulation with the aid of $\mathcal{C}_{i,j}$, except that: (1) the messages belonging to Stage-1 of the i-th session are relayed between the internal V^* and the external stand-alone knowledge-verifier; (2) whenever V^* (run internally by $\hat{P}^{(i,j)}$) successfully finishes, *for the first time*, Stage-1 of a session *w.r.t. an uncovered public key not in* $\mathcal{C}_{i,j}$, $\hat{P}^{(i,j)}$ just stops.

2. For the witness-extended simulator $E = (E_1, E_2)$ w.r.t. $\hat{P}^{(i,j)}$ and R_{KEY} [71], define the output of E_1 to be the messages received from $\hat{P}_{i,j}$ and the random coins used by S for simulating the prover-messages in Stage-1 of the i-th session in the main-thread. Then, invoke E_2 on input PK_j and run $\hat{P}^{(i,j)}$ to answer oracle queries to $\hat{P}^{(i,j)}$ from E_2.

3. In case SK_j was not extracted (i.e., E_2 outputs "$\perp$"), output "$\perp$" indicating failure. Otherwise, output SK_j. Note that the extraction-thread does not introduce any transcript into the main-thread.

Fig. 5. CZK simulation with main-thread and extraction threads.

and fails in covering the public key in question with only negligible probability (i.e., the knowledge error κ in Definition 2.7). What buried here by the witness-extended emulation lemma is the Goldreich-Kahan technique [42]: Specifically, the algorithm E_2 needs to first estimate the probability $p_{i,j}$ within a constant factor and then to use the estimated probability to ensure expected polynomial running time. As $p_{i,j} = p'_{i,j}$, E_2 can run $\hat{P}^{(i,j)}$ instead to estimate the probability $p_{i,j}$. Here, the fact that $p_{i,j} = p'_{i,j}$ is crucial to ensure expected polynomial-time CZK simulation.

As there are only at most $s(n)$ public keys to be covered in the simulation (i.e., only at most $s(n)$ extraction threads can be invoked), we have that S works in expected polynomial time. Also, by the witness-extended emulation lemma of [61], the probability that S outputs "$\perp$" (due to key-coverage failure in any of the invoked extraction threads) is also negligible. Then, the indistinguishability between the main-thread simulation output and the real interaction transcript is reduced to the (stand-alone) WI property of commit-then-SWI, by the following hybrid arguments.

For each $x_k \in \bar{\mathbf{x}}$, $1 \leq k \leq s(n)$, denote by w_k the corresponding $\mathcal{NP}$-witness such that $(x_k, w_k) \in R_L$. On input $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$ and the corresponding $\mathcal{NP}$-witnesses $\bar{\mathbf{w}} = (w_1, \ldots, w_{s(n)})$, and the public file $F = (PK_1, \ldots, PK_{s(n)})$, the ith hybrid experiment H_i, $1 \leq i \leq s(n)^2 + 1$, is defined as follows: Until the beginning of the ith Stage 2 (more precisely, the beginning of the ith commit-then-SWI) on a common statement $x_k \in \bar{\mathbf{x}}$ w.r.t. a public key PK_j,[11] where $1 \leq j, k \leq s(n)$, H_i acts just as the CZK simulator does (by invoking extraction threads). In particular, if the ith commit-then-SWI is reached w.r.t. an uncovered public key PK_j (i.e., V^* successfully finished the Stage 1 interactions w.r.t. an uncovered public key PK_j just prior to the beginning of the ith commit-then-SWI), H_i will first cover it by invoking an extraction thread. However, from the start of the ith Stage 2 and on, H_i works in a straight-line manner (without further invoking extraction threads) as follows: for all the remaining commit-then-SWI interactions (namely for any $\hat{i}$th commit-then-SWI, $\hat{i} \geq i$), H_i acts just as the honest prover does by using the $\mathcal{NP}$-witnesses in $\bar{\mathbf{w}}$;[12] but for any i'th commit-then-SWI, $i' < i$, that has not been completed up to the start of the ith commit-then-SWI, H_i still works as the simulator does by using the extracted secret key as the witness.[13] Clearly, the output of H_1 is statistically close to the real view of V^*, as the only difference between H_1 and the real interactions of V^* is that H_1 may potentially abort due to the first key-coverage failure which, however, occurs with negligible probability. On the other hand, the output of $H_{s(n)^2+1}$ is identical to the output of the simulator S. Supposing the output of the simulator S is distinguishable from the real view of V^*, there must exit an i, $1 \leq i \leq s(n)^2$, such that the output of H_i and that of H_{i+1} can be distinguishable. Note that the differences between H_i and H_{i+1} are as follows:

[11] Note that the ith commit-then-SWI (that is ordered by the occurrence of each Stage 2 message) may not necessarily be within the ith session (that is ordered by the occurrence of the first round, namely the first message of Stage 1, of each session).

[12] Note that if the ith commit-then-SWI is reached w.r.t. an uncovered public key PK_j, the corresponding secret key SK_j will be extracted but will never be used within the ith commit-then-SWI.

[13] Note that, according to the specification of the hybrid experiment H_i, for the i'th commit-then-SWI w.r.t. $PK_{j'}$, $i' < i$, $PK_{j'}$ must have been covered before the start of the ith commit-then-SWI.

Difference- 1. In H_i, the witness used in the ith commit-then-SWI (w.r.t. a common statement x_k and public key PK_j) is the true $\mathcal{NP}$-witness w_k, while in H_{i+1} the witness used in the ith commit-then-SWI is the extracted secret key SK_j.

Difference- 2. H_{i+1} may additionally abort due to secret key-coverage failure for the $(i+1)$th commit-then-SWI, in case the $(i+1)$th commit-then-SWI is reached w.r.t. an uncovered public key (i.e., V^* successfully finished the Stage 1 interactions w.r.t. an uncovered public key just before the start of the $(i+1)$th commit-then-SWI).

Now, consider another variant, denoted H'_i, which acts as H_i does, except that it uses the extracted secret key SK_j for the ith commit-then-SWI. It is clear that the output of H'_i and that of H_{i+1} are statistically close, as the only difference between H'_i and H_{i+1} is the above Difference 2 that can occur with negligible probability. Then, the indistinguishability between H_i and H'_i is reduced to the stand-alone WI property of commit-then-SWI, as follows. Specifically, we consider another algorithm $\hat{H}_i$, which mimics H_i but with the following modifications: The transcript of the ith commit-then-SWI (w.r.t. common statement x_k and public key PK_j) is generated by externally interacting with a commit-then-SWI prover $\hat{P}_i$, where the witness used by $\hat{P}_i$ on common input (x_k, PK_j) is either w_k or SK_j. Clearly, if $\hat{P}_i$ uses w_k (resp., SK_j) as its witness, the output of $\hat{H}_i$ is identical to that of H_i (resp., H'_i). Here, a key point is that from the start of the ith commit-then-SWI, the algorithm $\hat{H}_i$ does not invoke extraction threads any longer, and thus, the external interactions with $\hat{P}_i$ will never be rewound. Finally, a point of being worthy noting is that the normal WI property is defined with respect to probabilistic (strict) polynomial-time algorithms, but here $\hat{H}_i$ works in expected polynomial time. However, by Markov's inequality, it is easy to see that if the WI 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. A detailed treatment of this issue can also be found in [80].

(***Statistical***) **concurrent knowledge extraction.**

According to the CKE formulation, for any s-concurrent malicious prover P^* (cf. Sect. 2) we need to build a pair of algorithms (S, E). The simulator S, on input $(1^n, z)$, works as follows: It first perfectly emulates the key-generation stage of the honest verifier, getting $PK = (y_0, y_1)$ and $SK = s_b$ and $SK' = s_{1-b}$ for a random bit b. Then, S runs P^* on $(1^n, PK, R_L, z)$, where the random coins used by P^* is set by S. In the proof stages, S perfectly emulates the honest verifier with the secret key SK. Finally, whenever P^* stops, S outputs the simulated transcript str, together with the state information sta set to be (PK, SK, SK', z) and the random coins used by S. Note that the simulated transcript str is identical to the view of P^* in the real execution.

The knowledge extraction process is similar to that of [70]. Note that we need to extract the witnesses to all accepting sessions in str. Given (str, sta), the knowledge extractor E iteratively extracts witness for each accepting session. Specifically, for any i, $1 \leq i \leq s(n)$, we denote by E_i the experiment for the knowledge extractor on the ith session. E_i emulates S with the *fixed* random coins included in sta, with the exception that the random coins to be used by the simulator (emulating the honest verifier) for Stage 3 (i.e., SWIA/POK) of the ith session are no longer emulated internally, but received externally. The experiment E_i amounts to the execution of the SWIA/POK between a stand-alone (deterministic) prover and an honest verifier on common input

$(x_i, PK, c_w^{(i)})$, where $c_w^{(i)}$ is the Stage 2 message sent by P^* in the ith session. Suppose that the ith session w.r.t. common input x_i is accepting (otherwise, we do not need to extract a witness for that session and the witness is set to be "$\perp$"). According to the POK property of the underlying strong WIA/POK protocol, the knowledge extractor E can extract (w_i, r_i) in expected polynomial time with overwhelming probability.

Here, a subtle point needs to be further clarified. Denote by p the probability that E_i successfully finishes the SWIA/POK on input $(x_i, c_w^{(i)})$, by applying the (stand-alone) knowledge extractor on E_i, we get that the expected running time is $T(n) = p \cdot \frac{q(n)}{p-\kappa(n)}$, where $\frac{q(n)}{p-\kappa(n)}$ is the running time of the knowledge extractor and $\kappa(\cdot)$ is the knowledge error function (see Definition 2.7). However, when p is negligible, as clarified in [61], $T(n)$ may not be polynomial in n. The technique to deal with this issue is to apply the technique originally introduced in [42] or the more general witness-extended emulation lemma deliberated in [61]. The reader is referred to [42,61] for more details about the technique of dealing with this subtlety.

Now consider the value (w_i, r_i) extracted by E, which is fully determined by the single commitment $c_w^{(i)}$. Clearly such output necessarily falls into one of the following three cases:

Case- 1. $c_w^{(i)} = C(w_i, r_i)$ and $y_{1-b} = f(w_i)$. Recall that $PK = (y_0, y_1)$ and $SK = s_b$.
Case- 2. $c_w^{(i)} = C(w_i, r_i)$ and $y_b = f(w_i)$.
Case- 3. $c_w^{(i)} = C(w_i, r_i)$ and $(x_i, w_i) \in R_L$.

Case 1 can occur only with negligible probability, due to the one-wayness of f. Specifically, consider that y_{1-b} is given to the simulator as input, rather than being emulated internally. In more detail, let $y = f(s)$, where s is taken uniformly at random from $\{0, 1\}^n$ and is unknown to the simulator S and the extractor E. S generates PK as follows: It first takes a random bit $b \leftarrow \{0, 1\}$ and a random string $s' \leftarrow \{0, 1\}^n$, computes $y' = f(s')$, sets $y_b = y'$ and $y_{1-b} = y$, registers (y_0, y_1) as the public key and takes s' as the corresponding secret key. In this way, S still perfectly emulates the honest verifier; in particular, the Stage 1 interactions remain statistical WI to the malicious prover P^*. Then, supposing Case 1 occurs with non-negligible probability, E breaks the one-wayness of f also with non-negligible probability.

Case 2 can also occur with negligible probability, due to the statistical WI of Stage 1. Supposing Case 2 occurs with non-negligible probability (and we know Case 1 occurs with negligible probability), we can simply open $c_w^{(i)}$'s by brute-force to violate the statistical WI of Stage 1. In more detail, consider a brute-force algorithm B that runs P^* internally as a subroutine and mimics the algorithm S except that: all Stage 1 interactions on public key (y_0, y_1) are made by externally interacting with a statistical WI prover who uses s_b for a random bit $b \in \{0, 1\}$ as the witness. The goal of B is to guess the bit b with probability non-negligibly greater than $\frac{1}{2}$, which then violates the statistical WI property of Stage 1. It is clear that the view of P^* under the run of B is identical to that under the run of S. Supposing with probability p Case 2 occurs under the run of S, with the same probability it occurs under the run of B. After P^* stops, B randomly guesses an $i \in \{1, \ldots, s(n)\}$ and opens $c_w^{(i)}$ to w_i by brute-force. If $f(w_i) = y_\sigma$ for $\sigma \in \{0, 1\}$, the algorithm B just outputs σ; otherwise, it outputs a random bit. With probability at least

$\frac{p}{s(n)}$, Case 2 occurs w.r.t. the (randomly guessed) ith session, while w_i is the preimage of y_{1-b} only with negligible probability ϵ. It is straightforward to calculate that B correctly guesses b with probability at least $\frac{p}{s(n)} + \frac{1}{2}(1 - \frac{p}{s(n)} - \epsilon) = \frac{1}{2} + \frac{p}{2s(n)} - \frac{1}{2}\epsilon$, which violates the statistical WI property of Stage 1.

Remark. We use brute-force opening of $c_w^{(i)}$, rather than the output of the efficient knowledge extractor E_i, to obtain s_b and reach a contradiction with statistical WI of Stage 1. This is because E_i rewinds the prover, which, in turn, rewinds Stage 1 interactions. Since regular WI does not have to hold after rewinding, this will not lead to a contradiction. In more detail, by hybrid arguments, the security analysis of Case 2 can be reduced to the differences between the outputs of E_i in two respective hybrid experiments H_{k-1} and H_k, where $1 \le i, k \le s(n)$ and i is not necessarily equal to k. In the experiment H_k, the witness used in Stage 1 of the first k sessions is s_1 (i.e., the preimage of y_1), while that in the remaining $s(n) - k$ sessions is s_0. With non-negligible probability the witness extracted by the knowledge extractor E_i in H_{k-1} is s_0, and also with non-negligible probability the witness extracted by E_i in H_k is either s_1 or the witness w for $x \in L$ (but s_0 only with negligible probability). Suppose that the Stage 3 interactions of the ith session and the Stage 1 interactions of the kth session are interleaved, such that rewinding Stage 3 of the ith session incurs rewinding Stage 1 of the kth session. In such case, we do not know, to date, how to reach a contradiction with the statistical WI property of Stage 1 without brute-force opening of $c_w^{(i)}$. On the other hand, we notice that the above subtlety and obstacle might be surmounted, if a resettable WIAOK (as achieved in [17]) is applied in Stage 1.

By ruling out Case 1 and Case 2, now we conclude that for any i, $1 \le i \le s(n)$, if the ith session in str is accepting w.r.t. common input x_i selected by P^*, then E will output a witness w_i for $x_i \in L$. To finish the proof, we need to further show that knowledge extraction is independent of the secret key used by the simulator/extractor (i.e., the joint KEI property). Specifically, we need to show that $\Pr[R(SK, \bar{w}, str) = 1]$ is negligibly close to $\Pr[R(SK', \bar{w}, str) = 1]$ for any polynomial-time computable relation R, where $\bar{w}$ is the list of extracted witnesses (when the simulator/extractor uses SK as the witness in Stage 1 interactions in str) and SK' is the element (output by S in accordance with $\mathsf{Expt}_{\mathrm{CKE}}(1^n, z)$) randomly and independently distributed over the space of SK. The joint KEI property is direct from the *statistical* WI of Stage 1. Specifically, as the extracted witnesses are well defined by the statistically binding $c_w^{(i)}$'s, if the joint KEI property does not hold, we directly extract by brute-force all witnesses w_i's from $c_w^{(i)}$'s in successful sessions and then apply the assumed existing distinguishable relation R to violate the statistical WI of Stage 1.

In more detail, for any pair (s_0, s_1) in key-generation stage and for any auxiliary information z, $\Pr[R(SK, \bar{w}, str) = 1] = \frac{1}{2}\Pr[R(s_0, \bar{w}, str) = 1 | S/E$ uses s_0 in Stage 1 interactions in $str] + \frac{1}{2}\Pr[R(s_1, \bar{w}, str) = 1 | S/E$ uses s_1 in Stage 1 interactions in $str]$, and $\Pr[R(SK', \bar{w}, str) = 1] = \frac{1}{2}\Pr[R(s_0, \bar{w}, str) = 1 | S/E$ uses s_1 in Stage 1 interactions in $str] + \frac{1}{2}\Pr[R(s_1, \bar{w}, str) = 1 | S/E$ uses $s_0$in Stage 1 interactions$]$. Supposing the KEI property does not hold, it implies that there exists a bit $\alpha \in \{0, 1\}$ such that the difference between $\Pr[R(s_\alpha, \bar{w}, str) = 1 | S/E$ uses s_0 in Stage 1 interactions in str] and $\Pr[R(s_\alpha, \bar{w}, str) = 1|S/E$ uses $s_1$in Stage 1 interactions in str] is non-negligible. Now, we can incorporate the (s_α, R) into a brute-force algorithm in order to break the statistical WI of Stage 1. Further details are omitted here. Note that the KEI property holds against any (not necessarily polynomial-time computable) relation R. That is, the protocol depicted in Fig. 4 is of *statistical* CKE. □

On the essential role of strong WI.

We remark that, with respect to the above generic CZK–CKE implementation depicted in Fig. 4, the SWI at Stage 3 plays an essential role for achieving CZK and CKE properties simultaneously. In particular, we note that regular WI is insufficient here. On the one hand, we do not know how to prove the CZK property in general, when SWI is replaced by a regular WI. The underlying reason is that the composition of a (single) statistically binding commitment and regular WI is not necessarily to be regular WI. More detailed clarifications on this subtlety are given with the CZK analysis for the regular WI-based efficient CZK–CKE protocol in Sect. 7.1 (p. 43), where we employ double commitments to compose with regular WI for the CZK analysis and have to rely on complexity leveraging on one of the double commitments for the CKE analysis to go through. On the other hand, as ZK itself is SWI [38], one may suggest to use a special ZK (e.g., the FSZK which is composed of two regular WI sub-protocols) to replace SWI of Stage 3 such that the special ZK can share the regular WI of Stage 1 in the public-key model, and thus, we only use regular WIPOK at Stage 3. Recall that if we use FSZK as the underlying Stage 3 strong WI, the first WI sub-protocol of FSZK will be run on a pair of temporary keys (y'_0, y'_1), where for each $\sigma \in \{0, 1\}$, $y'_\sigma = f(t_\sigma)$ and $t_\sigma \in \{0, 1\}^n$. We want to remove this WI sub-protocol from FSZK, and just let the Stage 1 WI sub-protocol (that is run on the fixed key pair (y_0, y_1)) serve in addition as the first WI sub-protocol of FSZK. However, such a solution loses the CKE property and even concurrent soundness in general *in the public-key model* (see the concrete attack to FSZK in the public-key model [81]). That is, in the security analysis of the SWI-based generic CZK–CKE implementation, we have relied on the argument/proof of knowledge of SWI *in the plain model* that is not affected by concurrent composition in the plain model. If we replace the SWI by a ZK protocol *in the BPK model*, then we may require the ZK protocol have already been CKE secure, which, however, is our goal here.

Still, in Sect. 7, we shall consider more efficient CZK–CKE implementations based on regular WI. But the situation with such solutions turns out to be much subtler.

7. Efficient CZK–CKE in the BPK Model

In this section, we present the efficient constant-round CZK–CKE arguments for $\mathcal{NP}$ in the BPK model and the practical instantiations. The efficient CZK–CKE protocols rely on some minor complexity leveraging, in a novel way, to frustrate potential concurrent MIM. Along the way, we discuss and clarify the various subtleties.

Recall that for the generic CZK–CKE implementation presented in Sect. 6, the strong WI at Stage 3 plays an essential role for the provable security. However, employing strong WI complicates the protocol structure and incurs protocol inefficiency. It would be desirable to keep using regular WI at Stage 3, for conceptually simple protocol structure as well as for protocol efficiency. To bypass the subtleties of SWI for the CZK proof, we employ a double-commitment technique. Specifically, we require the prover to produce a *pair* of statistically binding commitments, c_w and c_{sk}, before starting the second WI sub-protocol, where c_w is supposed to commit to a valid $\mathcal{NP}$-witness for $x \in L$ and c_{sk} is supposed to commit to the preimage of either y_0 or y_1. Double commitments can bypass, by hybrid arguments, the subtleties of SWI for the CZK proof. However, the provable CKE property with double commitments turns out to be much subtler, and we have to employ (some minimal) complexity leveraging, in a novel way, to frustrate potential CMIM adversarial strategies. This renders us an efficient, as well as conceptually simple, CZK–CKE solution, which can further be practically instantiated for some number-theoretic languages.

The efficient construction is depicted in Fig. 6, p. 39. Though double commitments are used at Stage 2, the strong WIA/POK of Stage 3 in the generic construction is replaced by any regular WIA/POK here, from which much better efficiency advantage can be gained.

Notes on the complexity leveraging. We remark that complexity leveraging via the sub-exponential hardness assumption on verifiers' public keys is only for provable security analysis to frustrate concurrent MIM. Both CZK simulation and CKE knowledge

Key Generation. Let $f : \{0,1\}^n \rightarrow \{0,1\}^m$ be any OWF secure against 2^{n^c}-time adversaries for some constant c, $0 < c < 1$, where 1^n is the system security parameter. Each verifier V randomly and independently selects strings s_0, s_1 from $\{0,1\}^n$, randomly selects a bit $b \leftarrow \{0,1\}$, computes $y_b = f(s_b)$ and sets $y_{1-b} = f(s_{1-b})$. V registers $PK = (y_0, y_1)$ in a public file F as its public key, and keeps $SK = s_b$ as its secret key. Define $R_{KEY} = \{((y_0, y_1), s) | y_0 = f(s) \vee y_1 = f(s)\}$

Common input. An element $x \in L \cap \{0,1\}^{poly(n)}$, where L is any $\mathcal{NP}$-complete language. Denote by R_L the corresponding $\mathcal{NP}$-relation for L.

P's private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^{poly(n)}$ for $x \in L$. Here, we assume without loss of generality that the witness for any $x \in L \cap \{0,1\}^{poly(n)}$ is of the same length $poly(n)$.

Complexity leveraging. The system parameter is n, but the statistically-binding commitment c_{sk} is generated on a relatively smaller security parameter n_{sk}. Specifically, suppose the one-wayness of verifier's public key holds against 2^{n^c}-time adversaries for some constant c, $0 < c < 1$. Letting λ be any constant such that $\lambda > \frac{1}{c}$, we set $n = n_{sk}^{\lambda}$. Note that n and n_{sk} are still polynomially related. That is, any quantity that is polynomial in n is also another polynomial in n_{sk}, and vice versa. This complexity leveraging guarantees that although a $poly(n) \cdot 2^{n_{sk}}$-time adversary can break the hiding property of c_{sk} on the security parameter n_{sk}, it is still infeasible to break the one-wayness of f (because $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$).

Stage-1. V proves to P that it knows a preimage to one of y_0, y_1, by running a *statistical* WIA/POK protocol, in which V plays the role of knowledge prover. The witness used by V in this stage is s_b.

Stage-2. If V successfully finishes Stage-1, P does the following. P^* computes and sends $c_w = C(w, r_w)$ and $c_{sk} = C(0^n, r_{sk})$, where C is a statistically-binding commitment scheme and r_w and r_{sk} are the randomness used for commitments. c_{sk} is generated on the smaller security parameter n_{sk} specified above.

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

Fig. 6. Efficient CZK–CKE argument $\langle P, V \rangle$ for $\mathcal{NP}$ in the BPK model.

extraction are still of polynomial time. We note that the use of complexity leveraging for frustrating concurrent MIM could be a novel approach, different from the uses of complexity leveraging in existing works for protocols in the BPK model (e.g., [15]). Such an approach might also be applied to other scenarios to frustrate potential concurrent MIM strategies, while still providing polynomial-time simulation and/or knowledge extraction and retaining protocol efficiency and conceptually simple protocol structures. Note also that the complexity leveraging is minimal: It only applies to c_{sk} and all other components of the protocol work on the general system parameter n; also, all components except for verifiers' public keys can be standard polynomially secure. As we shall see in Sect. 7.3, the complexity leveraging can be waived as long as only concurrent soundness is concerned. We suggest that though non-standard, sub-exponential hardness assumption may still be viewed as reasonable, which is also used in a large body of works for fulfilling various cryptographic tasks. Detailed discussions on the subtleties surrounding the use of complexity leveraging are presented in Sect. 7.3.

On the necessity of *double* commitments c_w and c_{sk}. We stress that in the context of the protocol structure of efficient CZK–CKE depicted above in Fig. 6, mandating *double* commitments c_w and c_{sk} of Stage 2 plays a very crucial role for *simultaneously* achieving CZK and CKE in the public-key model. On the one hand, for protocol variants without either c_w or c_{sk}, concrete attacks exist, showing that they are not concurrently knowledge extractable (details are presented in Sect. 9). On the other hand, double commitments enable us to bypass the need of *strong* WI of Stage 3 for correct CZK simulation. Specifically, by employing double commitments the CZK simulation is not based on the strong WI property of Stage 3, and it is shown that regular WI is sufficient for correct CZK simulation by hybrid arguments.

Notes on the underlying hardness assumptions and round complexity. First note that except for the subexponential hardness assumption on the OWF f used in key generation, all other components for the efficient CZK–CKE protocol can be standard polynomially secure. If the statistically binding commitment C, used in Stage 2 and in Blum's protocol for DHC in Stage 3, is Naor's OWF-based scheme [65], its first-round initiation message can be merged with the last-round message from V in Stage 1 (and actually can be posted as a part of public key of V). Then, if the OWF f used in key generation admits statistical Σ protocols (and thus, we can use Σ_{OR} in Stage 1), the protocol depicted in Fig. 6 can be based on any sub-exponentially strong OWF admitting statistical Σ protocols and runs in six rounds. If we use in Stage 1 the modified Blum's protocol for DHC with constant-round statistically hiding commitments (cf. Section 2, p. 14), the protocol depicted in Fig. 6 runs in seven rounds, and is based either on any collision-resistant hash function together with any sub-exponentially strong OWF, or on any sub-exponentially strong claw-free collection (with efficiently recognizable index set). In the latter case (with modified Blum's protocol for DHC), we can use any sub-exponentially strong OWF for key generation.

7.1. *Practical Instantiations of CZK–CKE*

In order to get practical instantiations of the efficient CZK–CKE protocol depicted in Fig. 6 *without going through general $\mathcal{NP}$-reductions*, the verifier uses the sub-exponentially secure DLP-based OWF in key-generation stage: $f_{p,q,g}(s) = g^s \mod p$, where $s \leftarrow Z_q$, p and q are primes, $p = 2q + 1$ and $|p| = n$, and g is an element of $\mathbb{Z}_p^*$ of order q. We also assume the (standard polynomial-time) DDH assumption holds on the cyclic group indexed by (p, q, g), i.e., the sub-group of order q of $\mathbb{Z}_p^*$. The common input is (p, q, g, x), where $x \in \mathbb{Z}_p^*$ is of order q, and the corresponding witness is $w \in \mathbb{Z}_q$ such that $g^w = x \mod p$.

The statistical WIPOK of Stage 1 is replaced by the Σ_{OR} of Schnorr's basic protocol for DLP [76]. The perfectly binding commitment scheme of Stage 2 is replaced by the DDH-based ElGamal (non-interactive) commitment scheme [34] (cf. Sect. 2). To commit to a value $v \in \mathbb{Z}_q$, the committer randomly selects $u, r \in \mathbb{Z}_q$, computes $h = g^u \mod p$ and sends $(h, \bar{g} = g^r, \bar{h} = g^v h^r)$ as the commitment.

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

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

Again, by the Σ_{OR}-technique, if we have a practical Σ protocol of the first type, then we can also have a practical Σ protocol of the second type. Thus, to get the practical CZK–CKE implementation, all we need now is to develop a practical Σ protocol of the first type, which is referred to as Σ_{CTP} protocol (where "CTP" stands for "commit-then-proof") for presentation simplicity. Based on the Σ protocol for DLP [76], such a Σ_{CTP} protocol is described in Fig. 7 (p. 41).

We remark that, although the above practical implementation is for specific number-theoretic language, it is indeed very useful in practical scenarios. Next, we show that the Σ_{CTP} protocol depicted in Fig. 7 is indeed a Σ protocol, and have the following proposition.

Proposition 7.1. *The Σ_{CTP} protocol depicted in Fig. 7 is a Σ protocol.*

Proof. It is trivial to check that the completeness property holds. Below, we focus on the properties of special soundness and special honest-verifier zero knowledge.

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

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

Round-1: The prover P selects $t_0, t_1 \leftarrow \mathbb{Z}_q$ uniformly and independently, computes $a_0 = g^{t_0} \mod p$, $a_1 = g^{t_1} \mod p$ and $a_2 = h^{t_1} \mod p$, and sends (a_0, a_1, a_2) to the verifier V.

Round-2: V responds with a challenge e taken uniformly at random from $\mathbb{Z}_q$.

Round-3: P computes $z_0 = t_0 + we \mod q$ and $z_1 = t_1 + re \mod q$, and sends back (z_0, z_1) to V.

Verifier's decision: V accepts if $g^{z_0} = a_0 x^e \mod p$, $g^{z_1} = a_1 \bar{g}^e \mod p$ and $h^{z_1} = a_2(\bar{h}/x)^e \mod p$.

Fig. 7. Σ_{CTP} protocol based on DLP .

- Special soundness. From two accepting conversations w.r.t. the *same* Round 1 message, $((a_0, a_1, a_2), e, (z_0, z_1))$ and $((a_0, a_1, a_2), e', (z_0', z_1'))$, we can compute $w = \frac{z_0 - z_0'}{e - e'} \mod q$, and $r = \frac{z_1 - z_1'}{e - e'} \mod q$.
- Perfect SHVZK. The SHVZK simulator S works as follows: on common input $(x = g^w, \bar{g} = g^r, \bar{h} = g^w h^r = xh^r)$ and a given random challenge $e \in \mathbb{Z}_q$, it selects z_0, z_1 uniformly and independently from $\mathbb{Z}_q$, then it sets $a_0 = g^{z_0} x^{-e}$, $a_1 = g^{z_1} \bar{g}^{-e}$ and $a_2 = h^{z_1}(\bar{h}/x)^{-e}$, and outputs $((a_0, a_1, a_2), e, (z_0, z_1))$ as the simulated transcript. Denote by $\hat{t}_0 = z_0 - we$ and $\hat{t}_1 = z_1 - re$, we have that: $a_0 = g^{\hat{t}_0}, a_1 = g^{\hat{t}_1}$ and $a_2 = h^{\hat{t}_1}$. As z_0 and z_1 are taken uniformly and independently from $\mathbb{Z}_q$, we have that $\hat{t}_0$ and $\hat{t}_1$ are distributed uniformly and independently over $\mathbb{Z}_q$, from which the perfect SHVZK property follows. □

On practical instantiations of generic CZK–CKE. We observe that the same technique can also be applied to obtain a practical instantiation (without going through general $\mathcal{NP}$-reduction) of the general CZK–CKE protocol depicted in Fig. 4 (with the Σ_{OR}-based implementation of FSZK serving as the underlying SWI protocol in Stage 3) for the same DLP-based number-theoretic language. The resultant practical instantiation is based on the (polynomially secure) DDH assumption, but is more complex and less computationally efficient due to the use of Σ_{OR}-based implementation of FSZK. In more detail, by the Σ_{OR}-technique, practical instantiation of the generic CZK–CKE protocol is reduced to obtaining a practical Σ_{OR} protocol for the language L' defined in Fig. 4. When casted within the practical instantiation, on system parameters (p, q, g), $L' = \{(x = g^w, y_0, y_1, c_w = (h, \bar{g}, \bar{h})) | \exists (w, r) \ s.t. \ c_w = g^w h^r \wedge [x = g^w \vee (y_0 = g^w \vee y_1 = g^w)]\}$. Again, such a Σ_{OR} protocol for L' is reduced to the above Σ_{CTP} protocol.

7.2. *Security Analysis*

Theorem 7.1. *The protocol depicted in Fig. 6 is concurrently knowledge extractable CZK argument for $\mathcal{NP}$ in the BPK model.*

Proof. The completeness of the protocol $\langle P, V \rangle$ can be easily checked.

Concurrent zero knowledge.

For any $s(n)$-concurrent malicious verifier V^* (defined in Sect. 3) and any $\mathcal{NP}$-language L, the black-box CZK simulator S runs V^* as a subroutine on input $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$ (here, x_k might equal $x_{k'}$, $1 \leq k \neq k' \leq s(n)$) and the public file $F = (PK_1, \ldots, PK_{s(n)})$. S follows the main-thread/extraction thread approach as in the proof of Theorem 6.1 and works as follows: S acts just as the honest prover does in Stage 1 of any session. In Stage 2 of the ith session on a common input x_k and with respect to a public key PK_j, $1 \leq i \leq s(n)^2$ and $1 \leq k, j \leq s(n)$, if PK_j is still uncovered, by the witness-extended emulation lemma [61], S first invokes an extraction thread to successfully extract the secret key SK_j with overwhelming probability (otherwise, S aborts). After SK_j is extracted or PK_j has already been covered prior to Stage 2 of the

ith session, S computes $c_w^{(i)} = C(0^{poly(n)}, r_w^{(i)})$ and $c_{sk}^{(i)} = C(SK_j, r_{sk}^{(i)})$. Then, S runs the WIA/POK protocol with V^* in Stage 3 of the session with $(SK_j, r_{sk}^{(i)})$ as the witness. Finally, in case S does not abort (due to key-coverage failures), S outputs the transcript generated in the main-thread whenever V^* stops. Below, for presentation simplicity, we denote the combination of Stage 2 and Stage 3 of each session by "double-commitments-then-WI."

On input $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$, denote by $\bar{\mathbf{w}} = (w_1, \ldots, w_{s(n)})$ the corresponding $\mathcal{NP}$-witnesses such that for each k, $1 \leq k \leq s(n)$, $(x_k, w_k) \in R_L$. To show that the output of S is indistinguishable from the view of V^* in real concurrent interactions, we first consider another mental simulator M. M takes the witness list $\bar{\mathbf{w}}$ as an additional input and works just as S does, except that for any i, j and k, where $1 \leq i \leq s(n)^2$ and $1 \leq j, k \leq s(n)$, in Stage 2 of the ith session on common input x_k w.r.t. PK_j, M computes $c_w^{(i)} = C(w_k, r_w^{(i)})$, where w_k is the witness for the common input x_k. Note that the witness actually used by M in Stage 3 is still SK_j committed to $c_{sk}^{(i)}$, just as S does. That is, the $\mathcal{NP}$-witnesses committed to c_w's are never used by either S or M for Stage 3 interactions. The computational indistinguishability between the output of M and that of S is then from the computational hiding property of the underlying commitment scheme C used in Stage 2. Otherwise, by a simple hybrid argument, we can violate the hiding property of the commitment scheme C. Then, by another hybrid argument that is similar to that used in the proof of Theorem 6.1 (p. 33) but is actually more complicated due to the structure differences between commit-then-SWI and double-commitments-then-WI, the indistinguishability between the output of M and the real view of V^* is reduced to the regular WI property of Stage 3.

In more detail, on input $\bar{\mathbf{x}} = (x_1, \ldots, x_{s(n)}) \in L^{s(n)}$ and the corresponding $\mathcal{NP}$-witnesses $\bar{\mathbf{w}} = (w_1, \ldots, w_{s(n)})$, and the public file $F = (PK_1, \ldots, PK_{s(n)})$, the ith hybrid experiment H_i, $1 \leq i \leq s(n)^2 + 1$, is defined as follows. Until the beginning of the ith Stage 3[14] on a common statement $x_k \in \bar{\mathbf{x}}$ w.r.t. a public key PK_j, where $1 \leq j, k \leq s(n)$, H_i acts just as the above algorithm M does by invoking extraction threads and committing both the actual $\mathcal{NP}$-witnesses in $\bar{\mathbf{w}}$ and the extracted secret keys to Stage 2 messages. In particular, if PK_j was uncovered, H_i has covered it by invoking an extraction thread before the start of Stage 2 of that session. However, from the start of the ith Stage 3 and on, H_i works in a straight-line manner (without further invoking extraction threads) as follows:

- For all Stage 2 messages appearing after the staring of the ith Stage 3, H_i works as the honest prover does by committing an $\mathcal{NP}$-witness in $\bar{\mathbf{w}}$ (resp., 0^n) to the corresponding c_w (resp., c_{sk}).
- For all the remaining Stage 3 interactions (namely, for any $\hat{i}$th Stage 3, $\hat{i} \geq i$), H_i just uses the $\mathcal{NP}$-witnesses in $\bar{\mathbf{w}}$ (that is committed to the corresponding c_w) as the witness. Note that the Stage 2 message corresponding to the $\hat{i}$th Stage 3 may potentially be set before the start of the ith Stage 3. But for any i'th Stage 3, $i' < i$, that has not been completed up to the start of the ith Stage 3, H_i still works as

[14] Again, the ith Stage 3 (that is ordered by the occurrence of the first message of each Stage 3) may not necessarily be within the ith session (that is ordered by the occurrence of the first message of Stage 1 of each session).

M does by using the extracted secret key (that is committed to the corresponding c_{sk}) as the witness. Note that for the i'th Stage 3, $i' < i$, its corresponding Stage 2 message must have been set before the start of the ith Stage 3.

There are two differences between the output of H_1 and the real view of V^*: (1) For every Stage 2 message prior to the beginning of the first Stage 3 in H_1, the corresponding c_{sk} commits to an extracted secret key, while in the real view of V^* it commits to 0^n. However, the extracted secret keys committed to c_{sk}'s in H_1 are never used for Stage 3 interactions. Again, by the hiding property of the underlying commitment scheme, it can be shown by a simple hybrid argument that such a difference can only cause negligible distinguishability gap. (2) H_1 may potentially abort due to key-coverage failure before the beginning of the first Stage 3, which also occurs with negligible probability by the witness-extended emulation lemma [61]. Thus, the output of H_1 is computationally indistinguishable from the real view of V^*. On the other hand, the output of $H_{s(n)^2+1}$ is identical to the output of M. Supposing the output of the algorithm M is distinguishable from the real view of V^*, there must exist an i, $1 \leq i \leq s(n)^2$, such that the output of H_i and that of H_{i+1} are distinguishable. Note that the differences between H_i and H_{i+1} are:

Difference-1. In H_i, the witness used in the ith Stage 3 (w.r.t. a common statement x_k and public key PK_j) is the true $\mathcal{NP}$-witness w_k (committed to the corresponding c_w), while in H_{i+1} the witness used in the ith Stage 3 is the extracted secret key SK_j (committed to the corresponding c_{sk}).

Difference-2. H_{i+1} may additionally abort due to secret key-coverage failure between the start of the ith Stage 3 and that of the $(i+1)$th Stage 3.

Difference-3. For Stage 2 messages between the start of the ith Stage 3 and that of the $(i+1)$th Stage 3, the corresponding c_{sk}'s in H_i (resp., H_{i+1}) commit to 0^n (resp., the extracted secret keys).

Now, consider another variant, denoted H_i', which acts as H_i does, except that it uses the extracted secret key SK_j (committed to the corresponding c_{sk}) as the witness for the ith Stage 3. The output of H_i' and that of H_{i+1} differ on the above Difference 2 and Difference 3. Firstly, Difference 2 can occur with only negligible probability. For Difference 3, the key observation is that the values committed to these c_{sk}'s (between the start of the ith Stage 3 and that of the $(i+1)$th Stage 3) are never used for Stage 3 interactions in either H_i or H_{i+1}. This means that, with a simple hybrid argument, by the computational hiding property of the underlying commitment scheme the distinguishability gap between H_i' and H_{i+1} caused by Difference 3 is also negligible. Thus, the output of H_i' and that of H_{i+1} are computationally indistinguishable. Finally, the indistinguishability between H_i and H_i' is reduced to the WI property of the ith Stage 3, as follows. Specifically, we consider another algorithm $\hat{H}_i$ who generates (c_w, c_{sk}), where c_w (resp., c_{sk}) commits to w_k (resp., SK_j) and gives (c_w, c_{sk}) together with the corresponding decommitments to an external prover $\hat{P}_i$. $\hat{H}_i$ mimics H_i, except that the transcript for the ith Stage 3 (w.r.t. common statement x_k and public key PK_j and the Stage 2 message (c_w, c_{sk})) is generated by externally interacting with $\hat{P}_i$ on common input (c_w, c_{sk}), where the witness used by $\hat{P}_i$ is either w_k (committed to c_w) or SK_j (committed to c_{sk}). Clearly, if $\hat{P}_i$ uses w_k (resp., SK_j) as its witness, the output of $\hat{H}_i$

is identical to that of H_i (resp., H'_i). And the distinguishability between H_i and H'_i then violates the WI property of Stage 3.[15] Again, the fact that the algorithm $\hat{H}_i$ does not further invoke extraction threads from the start of the ith Stage 3 is critical to allow the above reduction, as it ensures that the external interactions with $\hat{P}_i$ will never be rewound.

(Statistical) concurrent knowledge extraction.

According to the CKE formulation, for any s-concurrent malicious prover P^* (defined in Sect. 2) we need to build two algorithms (S, E). The constructions of the algorithms (S, E) are almost identical to those in the proof of Theorem 6.1, and the reader is referred there for details. For each i, $1 \le i \le s(n)$, denote by (w_i, r_i) the output of E for the ith session that was accepting in the transcript str output by S. That output satisfies one of the following three cases.

Case- 1. $c_{sk}^{(i)} = C(w_i, r_i)$ and $y_{1-b} = f(w_i)$, where $c_{sk}^{(i)}$ and $c_w^{(i)}$ are the double statistically binding commitments sent at the Stage 2 of the ith session, C is the underlying statistically binding commitment scheme, and $SK = s_b$.

Case- 2. $c_{sk}^{(i)} = C(w_i, r_i)$ and $y_b = f(w_i)$.

Case- 3. $c_w^{(i)} = C(w_i, r_i)$ and $(x_i, w_i) \in R_L$.

As in the proof of Theorem 6.1, Case 1 can occur only with negligible probability, due to the one-wayness of f.

Proposition 7.2. *Case 2 occurs with negligible probability.*

Proof. (of Proposition 7.2). Before embarking on the actual proof, we first point out that the proof technique for ruling out Case 2 in the proof of Theorem 6.1 fails here. Specifically, in contrast to the proof of Theorem 6.1, brute-force opening of $c_{sk}^{(i)}$ and $c_w^{(i)}$ will, in general, not violate the statistical WI property of Stage 1 assuming Case 2 occurs with non-negligible probability. This will be well demonstrated by some exemplifying adversarial strategies to be presented and discussed in Sect. 7.3.1 (p. 45). Instead, we will use the statistical WI of Stage 1 and the complexity leveraging together to break the sub-exponential one-wayness of f in the subsequent proof. Also, as discussed in Sect. 7.2 (p. 37), we cannot use the output of the efficient knowledge extractor to reach a contradiction with WI of Stage 1 (and consequently waive the need of complexity leveraging), unless we employ a resettable WIAOK in Stage 1.

Supposing Case 2 occurs with non-negligible probability, this means that for some (s_0, s_1, b), where $s_0, s_1 \in \{0,1\}^n$ and $b \in \{0,1\}$, when the simulator S uses s_b as the

[15] Here, we remark that if only a single *statistically binding* commitment was used in Stage 2 (that commits to either w_k or sk_j), the above hybrid argument would fail in general, as the external prover $\hat{P}_i$ will be run on different statements for experiments H_i and H'_i, respectively. Specifically, for simulating the experiment H_i (resp., H'_i), the common input to $\hat{P}_i$ will be a commitment to w_k (resp., SK_j). In this case, the traditional WI property, which is defined w.r.t. a pair of witnesses to the same *fixed* common statement, cannot be used to derive any contradiction here. This is also the reason that we have employed strong WI for the generic CZK–CKE construction depicted in Fig. 4, where Stage 2 consists of only a single statistically binding commitment but the composition of commit-then-SWI ensures regular WI. We also remark that the problem is inherent to the composition of statistically binding commitment then WI. It seems that the composition of *statistically hiding* commitment then WI can still be WI.

witness for simulating Stage 1 interactions, with non-negligible probability $p(n)$, the $c_{sk}^{(i)}$ in the simulated transcript str output by S is a commitment of s_b. Otherwise, Case 2 will trivially occur with negligible probability. However, due to the statistical WI of Stage 1, with almost the same probability $p(n)$, the $c_{sk}^{(i)}$ in the simulated transcript str output by S, when it uses s_{1-b} as the witness for simulating Stage 1 interactions, is still a commitment of s_b. As in the proof of Theorem 6.1, the value committed to $c_{sk}^{(i)}$ can be brute-force extracted in time $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$. Now, supposing $y_b = f(s_b)$ is given to the simulator as input externally, and y_{1-b} and Stage 1 interactions are simulated by the simulator (with s_{1-b} as the witness), this implies that there exists an algorithm that can break the one-wayness of y_b in $poly(n) \cdot 2^{n_{sk}} \ll 2^{n^c}$-time with non-negligible probability, which violates the sub-exponential hardness of y_b. □

Remark. Notice that in the above proof, we relied on the string str produced by S in order to derive a contradiction. It was essential that str is (almost) independent of which secret key S uses, because Stage 1 is statistical WI, which then ensures that, in case Case 2 occurs, each of s_0 and s_1 is committed to str with non-negligible probability. Because str does not give us s_b explicitly, we have obtained it by brute-force in order to contradict the sub-exponential one-wayness of f via complexity leveraging. We would try to use the same approach of having S use the other secret key s_{1-b}, but instead of complexity leveraging, use knowledge extractor for E_i to obtain s_b. Unfortunately, this does not work in general, because the actual witness used by P^* in Stage 3, and thus the output of the knowledge extractor, may depend on which secret key S uses, and may not necessarily be s_b as expected but be the witness for $x_i \in L$ due to the double commitments used in Stage 2. Recall that CKE is considered w.r.t. the capability of convincing of even *true* statements (yet without "knowing" corresponding $\mathcal{NP}$-witnesses), in contrast to concurrent soundness only w.r.t. that of convincing of false statements. Specifically, though P^* commits s_b to str with non-negligible probability when S uses s_{1-b} as the witness of Stage 1, P^* may not use it as the witness of Stage 3 in this case; rather, P^* uses the witness w_i for $x_i \in L$ that is committed to $c_w^{(i)}$ as the witness of Stage 3, and then, the extracted witness will just be w_i (rather than s_b as expected) in this case. We do not know how to provably rule out such potential adversarial strategies without employing the complexity leveraging. Detailed clarifications of the subtleties are presented in Sect. 7.3, where we also show that the efficient CZK–CKE protocol depicted in Fig. 6 is still concurrently sound under standard polynomial-time hardness assumptions without complexity leveraging.

By ruling out Case 1 and Case 2, now we conclude that for any i, $1 \leq i \leq s(n)$, if the ith session in str is accepting w.r.t. common input x_i selected by P^*, then E will output a witness w_i for $x_i \in L$. To finish the proof, we need to further show that knowledge extraction is independent of the secret key used by the simulator/extractor (i.e., the joint KEI property). Specifically, we need to show that $\Pr[R(SK, \bar{w}, str) = 1]$ is negligibly close to $\Pr[R(SK', \bar{w}, str) = 1]$ for any polynomial-time computable relation R, where $\bar{w}$ is the list of extracted witnesses (when the simulator/extractor uses SK as the witness in Stage 1 interactions in str) and SK' is the element (output by S in accordance with $\mathsf{Expt}_{\mathrm{CKE}}(1^n, z)$) randomly and independently distributed over the space of SK. The joint

KEI property is direct from the *statistical* WI of Stage 1. Specifically, as the extracted witnesses are well defined by the statistically binding $c_w^{(i)}$'s, if the joint KEI property does not hold, we directly extract by brute-force all the witnesses w_i's from $c_w^{(i)}$'s of successful sessions and then apply the assumed existing distinguishable relation R to violate the statistical WI of Stage 1. The remaining analysis is the same as that in the proof of Theorem 6.1. □

7.3. *On the Subtleties without Complexity Leveraging*

In this section, we clarify the subtleties and justify the necessity of the (minimal) complexity leveraging on c_{sk} for the efficient CZK–CKE construction. We first discuss some concurrent man-in-the-middle (MIM) adversarial strategies potentially used by the malicious prover P^*, which we do not know how to provably rule out without employing the complexity leveraging. Then, we show that the efficient CZK–CKE protocol depicted in Fig. 7 is still concurrently sound under standard polynomial secure assumptions merely, i.e., without complexity leveraging.

7.3.1. *On the Use of Complexity Leveraging Against Man-in-the-Middle*

The key difficulty of ruling out Case 2 without complexity leveraging lies in the double commitments used in Stage 2 (and the fact that CKE is considered mainly for proving true statements). Specifically, to successfully finish the ith session on a *true* statement $x_i \in L$, for any i, $1 \leq i \leq s(n)$, an s-concurrent adversary P^* has *double* choices: It can use either the value committed to $c_{sk}^{(i)}$ or that committed to $c_w^{(i)}$ as the witness in Stage 3 regular WI. This is in sharp contrast to the commit-then-SWI-based generic CZK–CKE (depicted in Fig. 4), where to successfully finish the ith session P^* has to use the value committed to (determined by) the *unique* Stage 2 commitment $c_w^{(i)}$ as the witness in the Stage 3 SWI. To illustrate the subtlety, consider the following two potential adversarial strategies.

Adversarial Strategy-1. P^* commits a valid witness w (for $x_i \in L$) to $c_w^{(i)}$, and commits a secret key, say s_0, to $c_{sk}^{(i)}$ in Stage 2 of the ith session (*possibly by malleating verifier's public keys into x_i and $c_{sk}^{(i)}$*), where $x_i \in L$ is a true statement adaptively selected by P^* for the ith session. Then, *possibly by malleating the Stage 1 concurrent interactions*, P^* always uses the valid witness w in Stage 3 of the ith session in case the honest verifier V uses s_1 as the witness in Stage 1 interactions (*note that w could be maliciously related to s_1 as well, as the common input x_i is selected by P^**), but uses s_0 as the witness in Stage 3 in case V uses s_0 as the witness in Stage 1 interactions.

Adversarial Strategy-2. Suppose $SK = s_b$ where $b \leftarrow \{0, 1\}$; that is, V uses s_b as the witness during Stage 1 interactions. Then, depending on the bit b, P^* works as follows. On the one hand, with non-negligible probability p, P^* commits (w, s_b) to $(c_w^{(i)}, c_{sk}^{(i)})$ in Stage 2 of the ith session on a true statement $x_i \in L$ (*possibly by malleating verifier's public keys into $c_w^{(i)}$ and $c_{sk}^{(i)}$*)), where w is a valid witness for $x_i \in L$. Then, *possibly by malleating the Stage 1 concurrent interactions*, P^* successfully finishes Stage 3 of the session with s_b as the witness. On the other hand, with the same probability p, P^*

commits (w, s_{1-b}) to $(c_w^{(i)}, c_{sk}^{(i)})$ in Stage 2 of the ith session, but successfully finishes Stage 3 of the session with w as the witness.

Note that the concurrent malicious prover P^* actually amounts to a concurrent MIM who manages, by concurrent interleaving interactions, to malleate verifier's public keys and Stage 1 interactions (here, it plays the role of the verifier) into successful Stage 2 and Stage 3 interactions (here, it plays the role of the prover). Both of the above two exemplifying adversarial strategies indicate the failure of knowledge extraction correctness. Specifically, with non-negligible probability, the value extracted (when using $SK = s_b$ *for a random bit* b) is just the preimage of y_b committed to $c_{sk}^{(i)}$; that is, Case 2 in the CKE analysis of Theorem 7.1 occurs with non-negligible probability. However, no contradiction can be reached in order to rule out Case 2 without resorting to the complexity leveraging. In particular, they do not violate the statistical WI of Stage 1. For Adversarial Strategy 1, the values committed to $(c_w^{(i)}, c_{sk}^{(i)})$ are fixed. For Adversarial Strategy 2, with probability $2p$ the value committed to $c_w^{(i)}$ is w, and with probability p the value committed to $c_{sk}^{(i)}$ is s_0 (resp., s_1), no matter which secret key (whether s_0 or s_1) is used in the Stage 1 interactions. As a consequence, for both of the demonstrated adversarial strategies above, brute-force opening of $c_{sk}^{(i)}$ and $c_w^{(i)}$ does not break the statistical WI property of Stage 1.[16] As we do not employ any non-malleable building tools and we are actually facing a concurrent MIM P^*, the above MIM adversarial strategies could be potential. In general, we do not know how to *provably* rule out such seemingly impossible adversarial strategies, without resorting to the complexity leveraging.

We suggest that the use of complexity leveraging for frustrating concurrent MIM could be a novel approach, different from the uses of complexity leveraging in existing works (e.g., [15,80]). Such an approach may be possibly of independent interest and can be applied in other scenarios to frustrate potential concurrent MIM, while still providing polynomial-time simulation and/or knowledge extraction *as well as retaining the protocol efficiency and conceptually simple protocol structure*.

7.3.2. *Concurrent Soundness without Complexity Leveraging*

One key point allowing the above two potential concurrent MIM adversarial strategies is: P^* may use the $\mathcal{NP}$-witness w for $x_i \in L$ (committed to $c_w^{(i)}$) in Stage 3 interactions, which may potentially depend on which secret key S uses in Stage 1 interactions. However, if we are only concerned with concurrent soundness against P^*'s capability of convincing of a false statement "$x_i \in L$" (while $x_i \notin L$), the above two concurrent MIM strategies, actually Case 2 in the CKE analysis of Theorem 7.1, can be ruled out without employing the complexity leveraging.

In the following analysis, we only consider concurrent soundness, and assume no complexity leveraging on c_{sk}, i.e., verifier's public keys are standard polynomially secure and c_{sk} is computed on the same system parameter n. The analysis of concurrent soundness follows the same outline of CKE analysis in the proof of Theorem 7.1. Below, we only highlight the main differences between them. Supposing concurrent soundness does not

[16] This is in sharp contrast to the proof of Theorem 6.1, where brute-force opening of the statistically binding commitment $c_w^{(i)}$ is sufficient to violate statistical WI of Stage 1 if Case 2 occurs with non-negligible probability.

hold, then with non-negligible probability p there exists an i, $1 \leq i \leq s(n)$, such that an s-concurrent malicious prover P^* can convince of a false statement "$x_i \in L$" in the ith session, where, actually, $x_i \notin L$. By applying the stand-alone knowledge extractor on E_i (that uses s_b as the witness for Stage 1 interactions), we will get a witness, denoted (w_i, r_i), for the language L' defined for Stage 3 in Fig. 6. Here, the key observation is: as $x_i \notin L$, it must be the case that $c_{sk}^{(i)} = C(w_i, r_i)$ and w_i is the preimage of y_{1-b} (corresponding to Case 1) or y_b (corresponding to Case 2). In other words, supposing $x_i \notin L$, the commitment $c_w^{(i)}$ is meaningless for a successful run of the ith session. In this sense, the use of double commitments of Stage 2 is reduced to a situation similar to that of using only a single Stage 2 commitment in the generic CZK–CKE protocol depicted in Fig. 4, where we can rule out Case 2 without using the complexity leveraging nevertheless. Specifically, by the one-wayness of f, Case 1 occurs with negligible probability. Thus, Case 2 occurs (i.e., the extracted witness w_i is the preimage of y_b) with non-negligible probability. That is, with non-negligible probability the value committed to $c_{sk}^{(i)}$ indicates which witness is used in Stage 1. Just similar to the CKE analysis (p. 36) in the proof of Theorem 6.1, a brute-force algorithm can be constructed to violate the statistical WI of Stage 1. We have the following corollary:

Corollary 7.1. *The protocol depicted in Fig. 6 is concurrently sound in the BPK model without using the complexity leveraging, i.e., the one-way function f used in key generation is only standard polynomially secure and c_{sk} is computed on the same system parameter n.* □

7.4. *On the Necessity of Double Commitments*

For the efficient CZK–CKE protocol, as discussed, we have critically relied on the Stage 2 double commitments (namely c_w and c_{sk}), which are composed with the Stage 3 regular WI, in the CZK analysis (to avoid the subtleties surrounding the composition of a single statistically binding commitment and regular WI). On the other hand, the Stage 2 double commitments are the source of trouble for CKE analysis under merely standard polynomially secure assumptions, and we have relied on complexity leveraging to overcome the CKE proof obstacles.

In this section, we further show that, within the efficient CZK–CKE protocol structure, the Stage 2 double commitments are also essential for ensuring the CKE property. Specifically, by concrete attacks, we show that the protocol variant without c_w is not CKE secure (w.r.t. a language adaptively chosen based on verifiers' public keys); the variant without c_{sk} is even worse, which is not even concurrently sound (w.r.t. a language set statically and independently of verifiers' public keys). The attacks are similar to those described in Sect. 4, where a malicious prover P^* concurrently interacts with the honest verifier in two sessions and manages to malleate the Stage 1 interactions of one session into the Stage 2 and Stage 3 interactions in the other session. Details are presented in "Appendix".

8. Future Investigation

We conclude this work by proposing some questions for future investigation. In the future study, we are interested in fulfilling the following desired protocols:

- *Round-optimal* CZK–CKE in the BPK model.
- Efficient *black-box* CZK–CKE in the BPK model under standard complexity assumptions, without using complexity leveraging or strong WI. As discussed, this may call for the construction of efficient *black-box* resettable (statistical) WIAOK.
- Constant-round *adaptive input selection* concurrent ZK, and furthermore *concurrent non-malleable* ZK, with concurrent knowledge extraction in the BPK model.
- Constant-round *statistical* CZK with concurrent knowledge extraction in the BPK model. Notice that in this work, we achieve *computational* CZK with *statistical* CKE in the BPK model.
- CZK–CKE in the BPK model *merely from verifiers' public keys*. As discussed in Sect. 5.1, for black-box solutions we may require non-constant rounds or non-standard assumptions.
- CZK–CKE in the bounded player model.

Acknowledgments

First of all, we are grateful to the anonymous referees for their very helpful and insightful review comments and suggestions, which in particular have significantly improved this work. We are indebted to Oded Goldreich for many invaluable suggestions and discussions (particularly on strong WI and POK). We are grateful to Alessandra Scafuro and Ivan Visconti for many helpful discussions (particularly on round-optimal CZK in the BPK model) and for sending us an electronic copy of the work [75]. We thank Giovanni Di Crescenzo, Yehuda Lindell, Giuseppe Persiano and Alon Rosen for helpful discussions.

9. Appendix: On the Necessity of Double Commitments for Efficient CZK–CKE

To show the necessity of the double commitments c_w and c_{sk} used in Stage 2 of the efficient CZK–CKE protocol depicted in Fig. 6, we demonstrate concrete attacks against variants of the protocol without either c_w or c_{sk}, where WIA/POK protocols are implemented by Σ_{OR} protocols.

9.1. *The Attack Against Variant Protocol without c_w*

The variant protocol without c_w, which amounts to the CZK protocols of [24,82], is re-depicted in Fig. 8 (p. 54).

On the implementations of Σ_{OR}. For the Σ_{OR}-based protocol variant depicted in Fig. 8, to get *statistical* WI of Stage 1 there are two ways: In particular, we can require the underlying OWF f used in the key-generation stage admit perfect/statistical Σ protocols, and thus, the Σ_{OR} of Stage 1 is perfect/statistical WI. In general, the variant of (the n-parallel repetition of) Blum's protocol for DHC, where the statistically binding

Σ_{OR}-based protocol variant without c_w $\langle P, V \rangle$

Key Generation. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be any OWF where n is the security parameter. Each verifier V selects random strings s_0, s_1 from $\{0,1\}^n$, randomly selects a bit $b \leftarrow \{0,1\}$, computes $y_b = f(s_b)$ and sets $y_{1-b} = f(s_{1-b})$. V registers $PK = (y_0, y_1)$ in a public file F as its public key, and keeps $SK = s_b$ as its secret key.

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

P's private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^{poly(n)}$ for $x \in L$.

Stage-1. V proves to P that it knows the preimage of either y_0 or y_1, by running a Σ_{OR}-protocol on the input (y_0, y_1) in which V plays the role of the knowledge prover. The witness used by V in this stage is s_b. Denote by a_V, e_V and z_V, the first-, the second- and the third-round message of the Σ_{OR}-protocol, respectively.

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

Stage-3. Define a new $\mathcal{NP}$-language $L' = \{(x, y_0, y_1, c_{sk}) | [\exists w \ s.t. \ (x, w) \in R_L] \vee [\exists (w, r_{sk}, b) \ s.t. \ c_{sk} = C(w, r_{sk}) \wedge y_b = f(w) \wedge b \in \{0,1\}]\}$. Then, P proves to V that it knows a witness for $(x, y_0, y_1, c_{sk}) \in L'$, by running a Σ_{OR}-protocol (i.e., the OR-proofs of Σ-protocols). The witness used by P is w such that $(x, w) \in R_L$. We denote by a_P, e_P and z_P, the first-, the second-, and the third-round message of the Σ_{OR}-protocol of this stage, respectively.

Fig. 8. Σ_{OR}-based protocol variant without c_w.

commitments used in the first round are replaced by the one-round statistically hiding commitments based on collision-resistant hash functions, is a *statistical* Σ protocol (as well as statistical WI argument) for $\mathcal{NP}$, and thus can be applied to any $\mathcal{NP}$ language under the assumption of collision-resistant hash functions.

Let $\hat{L}$ be any $\mathcal{NP}$-language admitting a Σ protocol that is denoted by $\Sigma_{\hat{L}}$ (*in particular, $\hat{L}$ can be an empty set*). For an honest verifier V with its public key $PK = (y_0, y_1)$, we define a new language $L = \{(\hat{x}, y_0, y_1) \mid [\exists w \ s.t. \ (\hat{x}, w) \in R_{\hat{L}}] \vee [\exists (w, b) \ s.t. \ y_b = f(w) \wedge b \in \{0, 1\}]\}$. Note that for any string $\hat{x}$ (whether $\hat{x} \in \hat{L}$ or not), the statement "$(\hat{x}, y_0, y_1) \in L$" is always true as $PK = (y_0, y_1)$ is honestly generated. Also note that L is a language that admits Σ protocols (as Σ_{OR} protocol itself is a Σ protocol). Now, we describe the concurrent interleaving and malleating attack, in which P^* successfully convinces the honest verifier of the statement "$(\hat{x}, y_0, y_1) \in L$" for *any arbitrary $poly(n)$*-bit string $\hat{x}$ (*even if $\hat{x} \notin \hat{L}$*) by concurrently interacting with V in two sessions as follows.

1. P^* initiates the first session with V. After receiving the first-round message, denoted a'_V, of the Σ_{OR} protocol of Stage 1 of the first session on common input (y_0, y_1) (i.e., V's public key), P^* suspends the first session.
2. P^* initiates the second session with V and works just as the honest prover does in Stage 1 and Stage 2. We denote by c_{sk} the Stage 2 message of the second session

(i.e., c_{sk} commits to 0^n). When P^* moves into Stage 3 of the second session and needs to send V the first-round message, denoted a_P, of the Σ_{OR} protocol of Stage 3 of the second session *on common input* $(\hat{x}, y_0, y_1, c_{sk})$, P^* does the following:

- P^* first runs the SHVZK simulator of $\Sigma_{\hat{L}}$ (i.e., the Σ protocol for $\hat{L}$) on $\hat{x}$ to get a simulated conversation, denoted $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$." Then, P^* runs the SHVZK simulator of the underlying Σ protocol for $\mathcal{NP}$ on (y_0, y_1, c_{sk}) to get a simulated conversation, denoted (a_{sk}, e_{sk}, z_{sk}), for the (false) statement "$\exists(w, r_{sk}, b)\ s.t.\ c_{sk} = C(w, r_{sk}) \wedge y_b = f(w) \wedge b \in \{0, 1\}$."
- P^* sets $a_P = (a_{\hat{x}}, a'_V, a_{sk})$ and sends a_P to V as the first-round message of the Σ_{OR} protocol of Stage 3 of the second session, where a'_V is the one received by P^* in the first session.
- After receiving the second-round message of Stage 3 of the second session, denoted e_P (i.e., the random challenge from V), P^* sets $e'_V = e_P \oplus e_{\hat{x}} \oplus e_{sk}$ and then suspends the second session.

3. P continues the first session and sends $e'_V = e_P \oplus e_{\hat{x}} \oplus e_{sk}$ as the second-round message of the Σ_{OR} protocol of Stage 1 of the first session.
4. After receiving the third-round message of the Σ_{OR} protocol of Stage 1 of the first session, denoted z'_V, P^* suspends the first session again.
5. P^* continues the execution of the second session, and sends $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V), (e_{sk}, z_{sk}))$ to V as the last-round message of the second session.

Note that $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$ is an accepting conversation for the (possibly false) statement "$\hat{x} \in \hat{L}$," (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_0 or y_1, (a_{sk}, e_{sk}, z_{sk}) is an accepting conversation for the statement "$\exists(w, r_{sk}, b)\ s.t.\ c_{sk} = C(w, r_{sk}) \wedge y_b = f(w) \wedge b \in \{0, 1\}$," and furthermore, $e_{\hat{x}} \oplus e'_V \oplus e_{sk} = e_P$. According to the description of Σ_{OR} (presented in Sect. 2), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation of Stage 3 of the second session on common input $(\hat{x}, y_0, y_1)$. That is, P^* successfully convinced V of the statement "$(\hat{x}, y_0, y_1) \in L$" (*even for* $\hat{x} \notin \hat{L}$) in the second session *but without knowing any corresponding* $\mathcal{NP}$*-witness*.

9.2. *The Attack Against Variant Protocol without c_{sk}*

The variant protocol without c_{sk} is re-depicted in Fig. 9 (p. 56).

Now, we describe the concurrent interleaving and malleating attack, in which P^* successfully convinces the honest verifier of the statement "$x \in L$," for any n-bit string x and *for any* $\mathcal{NP}$*-language* L, without knowing any $\mathcal{NP}$-witness by concurrently interacting with V in two sessions as follows.

1. P^* initiates the first session with V. After receiving the first-round message, denoted a'_V, of the Σ_{OR} protocol of Stage 1 of the first session on common input (y_0, y_1) (i.e., V's public key), P^* suspends the first session.
2. P^* initiates the second session with V and works just as the honest prover does in Stage 1. In Stage 2 of the second session, P^* sends $c_w = C(0^n)$ (rather than $C(w)$ as honest prover does). When P^* moves into Stage 3 of the second session and

Σ_{OR}-based protocol variant without c_{sk} $\langle P, V\rangle$

Key Generation. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be any OWF, where n is the security parameter. Each verifier V selects random strings s_0, s_1 from $\{0,1\}^n$, randomly selects a bit $b \leftarrow \{0,1\}$, computes $y_b = f(s_b)$ and sets $y_{1-b} = f(s_{1-b})$. V registers $PK = (y_0, y_1)$ in a public file F as its public key, and keeps $SK = s_b$ as its secret key.

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

P's private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^n$ for $x \in L$. Here, we assume without loss of generality that the witness for any $x \in L \cap \{0,1\}^n$ is of the same length n.

Stage-1. V proves to P that it knows the preimage of either y_0 or y_1, by running a Σ_{OR}-protocol on the input (y_0, y_1) in which V plays the role of the knowledge prover. The witness used by V in this stage is s_b. Denote by a_V, e_V and z_V, the first-, the second- and the third-round message of the Σ_{OR}-protocol, respectively.

Stage-2. If V successfully finishes Stage-1, P computes $c_w = C(w, r_w)$, where C is a perfectly-binding commitment scheme and r_w is the randomness used for commitments.

Stage-3. Define a new $\mathcal{NP}$-language $L' = \{(x, y_0, y_1, c_w) | (\exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge (x, w) \in R_L) \vee (\exists (w, b)\ s.t.\ y_b = f(w) \wedge b \in \{0,1\})\}$. Then, P proves to V that it knows a witness for $(x, y_0, y_1, c_w) \in L'$, by running a Σ_{OR}-protocol. The witness used by P is (w, r_w). We denote by a_P, e_P and z_P, the first-, the second-, and the third-round message of the Σ_{OR}-protocol of this stage, respectively.

Fig. 9. Σ_{OR}-based protocol variant without c_{sk} .

needs to send V the first-round message, denoted a_P, of the Σ_{OR} protocol of Stage 3 of the second session *on common input* (x, y_0, y_1, c_w), P^* does the following:

- P^* first runs the SHVZK simulator of the underlying Σ protocol for $\mathcal{NP}$ on common input (x, c_w) to get a simulated conversation, denoted (a_x, e_x, z_x), for the (false) statement "$\exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge (x, w) \in R_L$)."
- P^* sets $a_P = (a_x, a'_V)$ and sends a_P to V as the first-round message of the Σ_{OR} protocol of Stage 3 of the second session, where a'_V is the one received by P^* in the first session.
- After receiving the second-round message of Stage 3 of the second session, denoted e_P (i.e., the random challenge from V), P^* sets $e'_V = e_P \oplus e_x$ and then suspends the second session.

3. P continues the first session and sends $e'_V = e_P \oplus e_x$ as the second-round message of the Σ_{OR} protocol of Stage 1 of the first session.
4. After receiving the third-round message of the Σ_{OR} protocol of Stage 1 of the first session, denoted z'_V, P^* suspends the first session again.
5. P^* continues the execution of the second session and sends $z_P = ((e_x, z_x), (e'_V, z'_V))$ to V as the last-round message of the second session.

Note that (a_x, e_x, z_x) is an accepting conversation for the (false) statement "$\exists (w, r_w)\ s.t.$ $c_w = C(w, r_w) \wedge (x, w) \in R_L$)," (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_0 or y_1, and furthermore, $e_x \oplus e'_V = e_P$. According to the description of Σ_{OR} (cf. Sect. 2), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation of Stage 3 of the second session on common input x; that is, P^* successfully convinced V of the statement "$x \in L$" *but without knowing any corresponding $\mathcal{NP}$-witness*. The above attack also indicates that the protocol variant without c_{sk} is not even concurrently sound.

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, R. Canetti, J. B. Nielsen and R. Pass. UniversallyComposable Protocols with Relaxed Set-Up Assumptions. In *IEEESymposium on Foundations of Computer Science*, pages 186–195, 2004.

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

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

[5] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation andExtraction. *SIAM Journal on Computing*, 33(4): 783–818, 2004.

[6] B. Barak, Y. Lindell and S, Vadhan. Lower Bounds for Non-Black-Box Zero-Knowledge. *Journal of Computer and System Sciences*, 72(2): 321–391, 2006.

[7] B. Barak, M. Prabhakaran, and A. Sahai. Concurrent Non-Malleable Zero-Knowledge. FOCS 2006: 345–354.

[8] 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.

[9] M. Bellare and O. Goldreich. On Probabilistic versus Deterministic Provers in the Definition of Proofs Of Knowledge. Electronic Colloquium on Computational Complexity, 13(136), 2006. A slightly refined version also appears in [47], pages 114–123, 2011.

[10] M. Bellare, R. Impagliazzo and M. Naor. Does Parallel RepetitionLower the Error in Computationally Sound Protocols? In *IEEESymposium on Foundations of Computer Science*, pages 374–383, 1997.

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

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

[13] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. *Eurocrypt* 1998: 59–71.

[14] G. Brassard, D. Chaum and C. Crepeau. Minimum Disclosure Proofsof Knowledge. *Journal of Computer Systems and Science*, 37(2):156–189, 1988.

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

[16] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-BoxConcurrent Zero-Knowledge Requires (Almost) Logarithmically ManyRounds. In *SIAM Journal on Computing*, 32(1): 1–47, 2002.

[17] C. Cho, R. Ostrovsky, A. Scafuro and I. Visconti. Simultaneously Resettable Arguments of Knowledge. TCC 2012: 530–547.

[18] K. M. Chung, R. Ostrovsky, R. Pass, M. Venkitasubramaniam and I. Visconti. 4-Round Resettably-Sound Zero Knowledge. TCC 2014: 192–216.

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

[20] 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 893*, pages 174–187. Springer-Verlag, 1994.

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

[22] I. Damgard. Lecture Notes on Cryptographic Protocol Theory. BRICS, Aarhus University, 2003. Available from: http://www.daimi.au.dk/~ivan/CPT.html

[23] I. Damgard, T. Pedersen and B. Pfitzmann. On the Existence of Statistically-Hiding Bit Commitment and Fail-Stop Signatures. In CRYPTO 1993: 250–265.

[24] Y. Deng and D. Lin. Resettable Zero Knowledge in the Bare Public Key Model under Standard Assumption. Inscrypt 2007, pages 123–137.

[25] Y. Deng, D. Feng, V. Goyal, D. Lin, A. Sahai and M. Yung. Resettable Cryptography in Constant Rounds: the Case of Zero Knowledge. Asiacrypt 2011, pages 390–406. Available also from Cryptology ePrint Archive, Report No. 2011/408.

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

[27] 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.

[28] 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.

[29] G. Di Crescenzo and I. Visconti. Personal communications, 2004.

[30] G. Di Crescenzo and I. Visconti. On Defining Proofs of Knowledge in the Bare Public Key Model. In *Italian Conference on Theoretical Computer Science (ICTCS)*, 2007.

[31] 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.

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

[33] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *H. Krawczyk (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1998, LNCS 1462*, pages 442–457. Springer-Verlag, 1998.

[34] T. El Gamal. A Public Key Cryptosystem and Signature Scheme Basedon Discrete Logarithms. *IEEE Transactions on InformationTheory*, 31: 469–472, 1985.

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

[36] 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.

[37] U. Feige and A. Shamir. Witness Indistinguishability and WitnessHiding Protocols. In *ACM Symposium on the Theory ofComputing*, pages 416–426, 1990.

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

[39] O. Goldreich. *Foundations of Cryptography-Basic Applications*. Cambridge University Press, 2002.

[40] O. Goldreich. *Studies in Complexity and Cryptography*. LNCS 6650, Springer-Verlag, 2011.

[41] O. Goldreich. *Strong Proofs of Knowledge*. Pages 55–59 in [47].

[42] 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.

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

[44] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design. In *IEEE Symposium on Foundations of Computer Science*, pages 174–187, 1986.

[45] O. Goldreich, S. Micali and A. Wigderson. How to Prove all $\mathcal{NP}$-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In *A. M. Odlyzko (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1986, LNCS 263,* pages 104–110, Springer-Verlag, 1986.

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

[47] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All languages in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery,* 38(1): 691–729, 1991. Preliminary version appears in [51, 52].

[48] 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.

[49] S. Goldwasser, S. Micali and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks. *SIAM Journal on Computing*, 17(2): 281–308, 1988.

[50] V. Goyal, A. Jain, R. Ostrovsky, S. Richelson and I. Visconti. Concurrent Zero Knowledge in the Bounded Player Model. TCC 2013: 60–79.

[51] V. Goyal, A. Jain, R. Ostrovsky, S. Richelson and I. Visconti. Constant-Round Concurrent Zero Knowledge in the Bounded Player Model. Asiacrypt 2013: 21–40.

[52] 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.

[53] J. Hastad, R. Impagliazzo, L. A. Levin and M. Luby. Construction of a Pseudorandom Generator from Any One-Way Function *SIAM Journal on Computing*, 28(4): 1364–1396, 1999.

[54] H. Hastad, R. Pass, D. Wikstrom and K. Pietrzak. An Efficient Parallel Repetition Theorem. In TCC 2010, pages 1–18, 2010.

[55] I. Haitner and O. Reingold. Statistically-Hiding Commitment from Any One-Way Function. STOC 2007: 1–10.

[56] I. Haitner, O. Horvitz, J. Katz, C. Koo, R. Morselli and R. Shaltiel. Reducing Complexity Assumptions for Statistically-Hiding Commitments. In Eurocrypt 2005: 58–77.

[57] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In CRYPTO 1996: 201–215.

[58] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in polyloalgorithm rounds. In STOC, pages 560–569, 2001.

[59] 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.

[60] H. Lin and R. Pass. Constant-Round Non-Malleable Commitments from Any One-Way Function. STOC 2011: 705–714.

[61] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology*, 16(3): 143–184, 2003. Preliminary version appeared in CRYPTO 2001.

[62] Y. Lindell. Constant-Round Zero-Knowledge Proof of Knowledge. ECCC Report No. 2011/003.

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

[64] 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.

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

[66] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Perfect Zero-Knowledge Arguments for NP Using Any One-Way Permutation. *Journal of Cryptology*, 11(2): 87–108, 1998.

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

[68] R. Ostrovsky, G. Persiano and I. Visconti. Constant-Round Concurrent Non-malleable Zero Knowledge in the Bare Public Key Model. *ICALP(2) 2008, LNCS 5126*, pages 548–559, 2008. Full version available from ECCC Report No. 2006/095.

[69] R. Pass, W.-L. Dustin Tseng, and M. Venkitasubramaniam: Concurrent Zero Knowledge, Revisited. *Journal of Cryptology*, 27(1): 45–66, 2014.

[70] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. SIAM Journal on Computing, 37(6): 1891–1925 (2008). Preliminary version appears in In *IEEE Symposium on Foundations of Computer Science*, pages 563–572, 2005.

[71] R. Pass and M. Venkitasubramaniam. An Efficient Parallel Repetition Theorem for Arthur-Merlin Games. In *ACM Symposium on Theory of Computing*, pages 420–429, 2007.

[72] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In FOCS, pages 366–375, 2002.

[73] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In Eurocrypt, pages 415–432, 1999.

[74] P. Rogaway. Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. Vietcrypt 2006, LNCS 4341, pages 221–228.

[75] A. Scafuro and I. Visconti. On Round-Optimal Zero Knowledge in the Bare Public Key Model. Eurocrypt 2012, LNCS 7237, pages 153–171.

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

[77] I. Visconti. Efficient Zero Knowledge on the Internet. *ICALP 2006, LNCS 4052*, pages 22–33, Springer-Verlag.

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

[79] A. Yao, M. Yung and Y. Zhao. Concurrent Knowledge Extraction in the Public Key Model. ICALP 2010, Part I, LNCS 6198, pages 702–714, 2010. Preliminary version appears in Electronic Colloquium on Computational Complexity (ECCC), Report No. 2007/002.

[80] M. Yung and Y. Zhao. Generic and practical resettable zero-knowledge in the bare public key model. In *M. Naor (Ed.): Advances in Cryptology-Proceedings of Eurocrypt 2007, LNCS 4515*, pages 116–134, Springer-Verlag, 2007. Preliminary version appears in ECCC Report No. 2005/048.

[81] M. Yung and Y. Zhao. Interactive Zero-Knowledge with Restricted Random Oracles. In *S. Halevi and T. Rabin (Ed.): Theory of Cryptography (TCC) 2006, LNCS 3876*, pages 21–40, Springer-Verlag, 2006.

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