Electronic Colloquium on Computational Complexity, Report No. 2 (2007)

Concurrent Knowledge Extraction in the Public-Key Model

Moti Yung[*] Yunlei Zhao[†]

Abstract

Knowledge extraction is a fundamental notion, modeling knowledge possession in a computational complexity sense. The notion provides a tool for cryptographic protocol design and analysis, enabling one to argue about the internal state of protocol players. We define and investigate the relative power of the notion of "concurrent knowledge-extraction" (CKE) in the concurrent zero-knowledge (CZK) bare public-key (BPK) model, namely we investigate how to formally treat knowledge possessions for parties interacting over the Internet (say). We further investigate the implementation of a generic scheme for this new notion of CKE concurrent zero-knowledge (CZK-CKE) arguments for $\mathcal{NP}$ in this model.

Concurrent knowledge-extraction in the public-key model essentially means that for any $\mathcal{NP}$ statement whose validation is successfully conveyed by a possibly malicious prover to an honest verifier (with registered public-key) employing concurrent interactions, the prover "must know" the corresponding witness. It is shown that under any one-way function (OWF), concurrent knowledge-extraction is strictly stronger than concurrent soundness in the BPK model (as is demonstrated by concrete attacks). Then, in light of our concurrent interleaving and malleating attacks, we formalize CKE in the public-key model.

We then present, both general scheme (round-optimal or minimal hardness assumption based for $\mathcal{NP}$) and practical scheme (based on the DDH assumption for specific number-theoretic language). Both schemes are constant-round CZK-CKE arguments in the BPK model. We note that both the ZK simulation and the knowledge extraction in our model and proofs are efficient (i.e., expected polynomial-time).

Then, we discuss an extended notion of CKE, called joint CKE (JCKE), which essentially requires that the malicious prover "knows" the corresponding *joint* witnesses to all statements successfully convinced in its concurrent interactions. We show that our practical CZK-CKE scheme also satisfies this seemingly stronger notion, and further show how to slightly modify the general scheme to satisfy this extended notion as well.

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

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

ISSN 1433-8092

1 Introduction

Zero-knowledge (ZK) protocols allow a prover to validate theorems to a verifier without giving away any other knowledge other than the theorems being true (i.e., existing witnesses). This notion was introduced by Goldwasser, Micali and Rackoff [36] and its generality was demonstrated by Goldreich, Micali and Wigderson [35]. Since its introduction ZK has found numerous and extremely useful applications, and by now has been playing the central role in modern cryptography.

The concept of "proof of knowledge (POK)" was informally introduced in [36], and was formally treated in [5, 32]. POK systems, especially zero-knowledge POK (ZKPOK) systems, play a fundamental role in the designing of cryptographic schemes and protocols, and enable a formal complexity theoretic treatment of what does it mean for a machine to "know" something. Very roughly, by "proof of knowledge" we mean that a possibly malicious prover can convince that an $\mathcal{NP}$ statement is true if and only if it, in fact, "knows" (i.e., possesses) a witness to the statement (rather than only convincing the language membership of the statement, i.e., the fact that a corresponding witness exists).

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 are run concurrently at the same time, studying security properties of ZK protocols in such concurrent settings has attracted extensive research efforts in recent years, initiated by Dwork, Naor and Sahai [26]. Informally, a ZK protocol is called concurrent zero-knowledge (CZK) if the ZK related simulatability property holds in the concurrent settings, namely, when a malicious verifier concurrently interacts with a polynomial number of honest prover instances and schedules message exchanges as it wishes.

A major measure of efficiency for interactive protocols is the round-complexity. Unfortunately, there are no constant-round CZK protocols in the standard model, at least for the black-box case, as implied from the work of Canetti, Killian, Petrank and Rosen [11]. To get constant-round concurrent ZK protocols, several setup models have been introduced: the timing model [26, 27], the preprocessing model [21], the common reference string model [15], the certified public-key infrastructure [2] and the bare public-key model [10], etc.

A protocol in the BPK model, introduced by Canetti, Goldreich, Goldwasser and Micali [10], simply assumes that all verifiers have deposited a public key in a public file before (or while) the interaction takes place among the users. Note that, 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) [10]. That is, no trusted third party is assumed, and preprocessing is reduced to users non-interactively posting public-keys in a public file [23]. 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 later added key certification is assumed). It was pointed out by Micali and Reyzin [46] that BPK is, in fact, applicable to interactive systems in general.

Soundness (i.e., verifier security) in the BPK model turned out to be much more complicated and subtle than otherwise, as was shown by Micali and Reyzin [46]. They showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger: one-time, sequential, concurrent and resettable soundness. In this work, we focus on concurrent soundness, which roughly means that a malicious prover P^* cannot convince the honest verifier V of a *false* statement even when P^* is allowed multiple interleaving interactions with V in the public-key model. Micali and Reyzin also showed that any black-box ZK protocols with concurrent soundness in the BPK model (for non-trivial languages outside $\mathcal{BPP}$) must run at least four rounds [46]. It is also shown in [3, 46] that *black-box* ZK arguments with resettable soundness only exist for trivial (i.e, $\mathcal{BPP}$) languages (whether in the BPK model or not).

Due to the above, it was implied that concurrent soundness might be the best verifier security one can hope for black-box ZK arguments in the BPK model. In this work, we show that this intuition is not entirely correct, at least not in the setting of proof of knowledge where provers are polynomial time. Specifically, concurrent soundness only guarantees that concurrently interleaved interactions cannot help a malicious prover to convince the honest verifier of a *false* statement in the public-key model. But, it does *not* prevent a malicious prover from convincing the honest verifier of a *true* statement but without knowing any witness for the statement being proved. One reason that this potential vulnerability is not merely a theoretical concern is that all 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. *Further, this type of proofs are also quite common in practical cryptographic protocols.* Thus, a malicious prover can potentially malleate the verifier's interactions in one session into successful interactions in another concurrent session on a true (e.g., verifier's public-key related) statement but without knowing any witness for the statement being proved. We show that such potential vulnerability turns out to be a real security threat for existing cryptographic protocols concurrently run in the public-key model. This motivates the need for careful definition and for achieving concurrent verifier security for concurrent ZK protocols in the BPK model, so that provably the security vulnerability in proof of knowledge is frustrated.

1.1 Our contributions

We start by investigating the subtleties of concurrent verifier security in the public-key model in the case of proof of knowledge. Specifically, we show a concurrent interleaving and malleating attack against the concurrent ZK protocol of [23]. This actual protocol is, both, concurrently sound and normal (stand-alone) argument of knowledge in the BPK model. This shows that concurrent soundness and normal arguments of knowledge do not guarantee concurrent verifier security. (It is also further clarified recently [58] that the formulations of concurrent non-malleability (CNM) in the public-key model in exiting works [50, 20] do not capture CKE in the public-key model, *which further shows the subtleties of correctly formulating CKE in the public-key model.*)

Then, we formulate concurrent verifier security that frustrates the vulnerability demonstrated by the attack which is of the man-in-the-middle type. The security notion defined is called `concurrent knowledge-extraction (CKE)` in the public-key model, which essentially means that for any statement that is successfully proved by a possibly malicious prover to an honest verifier (with registered public-key) by concurrent interactions, the prover must "know" the corresponding witness.

We then design, both general (round-optimal or minimal hardness assumption optimal) construction and a practical construction (based on the DDH assumption), which are constant-round CZK-CKE arguments in the BPK model.

Finally, we discuss an extended notion of CKE, called joint CKE (JCKE), which essentially requires that the malicious prover "knows" the corresponding *joint* witnesses to all statements successfully convinced in its concurrent interactions. We show that a slight modification of the general construction as well as the practical construction also satisfy this seemingly stronger notion.

1.2 Related Works

Let us review some recent results and developments; we have been involved in numerous recent works which we review together with related works. While the list of related works and related issues is quite lengthy, the bottom line is that the notion defined and achieved herein is unique and independent of various related issues and works, and it captures knowledge extraction as a basic issue in concurrent executions in public key models.

Concurrent ZK (actually, resettable ZK that is stronger than CZK) arguments for $\mathcal{NP}$ with a *provable* CKE property in the BPK model was first achieved in our precursory unpublished work [56]. However, the CKE property of the protocol of [56] is achieved under sub-exponential hardness assumptions, and thus holds only for sub-exponentially hard languages. CKE for concurrent ZK arguments in the BPK model under standard assumptions were left over there as an open problem, which we answer here. The subtleties of concurrent knowledge-extraction were also previously considered in [24]. But, the work of [24] did not realize the crucial issue of "knowledge-extraction independence" formulated in this work (driven by concrete attacks on natural existing works). The CKE formulation formulated in [24] could be viewed as a natural extension of the normal arguments of knowledge into the public-key model. Also, [24] did not note the relationship between CKE and concurrent soundness in the public-key model, and achieving CKE for concurrent ZK in the BPK model was left over there as an open problem.

Two constructions for concurrent ZK arguments with sequential soundness in the BPK model under standard assumptions were proposed in the incomplete work of [59] (the early version since January 2004). But, the security proof of concurrent soundness turned out to be flawed, as observed independently in [23, 57]. One construction was fixed to be concurrently sound in [23] by introducing some key techniques, and recently another construction was fixed to be concurrently sound in [19] following the spirit of [23]. Given these works, the current work further shows that the concurrently sound CZK arguments of [23, 19] do not capture CKE and are not concurrently knowledge-extractable when it comes to proofs of knowledge.

Recently it is clarified in [58] that the formulations of concurrent non-malleability (CNM) in exiting works [50, 20] do not capture CKE in the public-key model. For the sake of clear understanding of the difference between these notions and the one in the current paper, the clarifications from [58] are re-presented in Appendix A. We note that the preliminary version of this work appears in parts of [56] and the incomplete work [59] (the versions since 19 July 2006) that are independent of [50, 20]. The works of [50, 20] deal with concurrent man-in-the-middle (CMIM) adversaries in the *authenticated* BPK model (which is stronger than the BPK model and is shown to be necessary for constant-round CNMZK in the public-key model [20]). Again, concrete attacks are presented in [58] on the protocol of [20], showing that it is not concurrently knowledge-extractable. In comparison with the work of [50], we remark that [50] and this work are incomparable. The work of [50] deals with concurrent non-malleability, formulates and achieves constant-round concurrently non-malleable witness-indistinguishability (CNMWI) arguments *in the plain model* under any collision-resistant hash function, with CNMZK and the more general secure multi-party computing in the (authenticated) BPK model as major applications. This is achieved in [50] by critically using the recent breakthrough non-black-box techniques from [1, 52, 53]. Again, note that the CNM formulation in [50] does not capture the CKE formulated in this work; in addition it does not concentrate on protocols of optimal rounds or practical efficiency. More detailed comparison can be found in Appendix A drawn from [58].

A reasonable amount of works investigated resettable ZK in the BPK model. Looking forward, we note that the CZK-CKE arguments presented in this work might in principle be modified into rZK arguments with *non-black-box* CKE in the BPK model under any collision-resistant hash function (using non-black-box techniques from [1, 52, 53] similar to that of [19]), but at the price of losing efficiency and hardness generality (*even if it could be shown to be a correct direction in the future*), and thus it will be incomparable with the current work. This research line is outside the scope of the current work.

We do not deal with concurrent non-malleability nor resettable ZK in this work, but we remark that CZK-CKE are themselves useful fundamental tools and can be used in other applications (they are the concurrent version of the highly useful ZK arguments of knowledge in the BPK model). Also, we suggest that the clarifications and formulation of CKE in the public-key model presented in this work are of independent value and can serve as a basis for formulating and achieving more complex interactive cryptographic protocols in the public-key model.

We remark that in [31] Garay and MacKenzie noted that when argument/proof of knowledge protocols are used as building block in a larger protocol in the plain model, concurrent nested rewinding and interleaving may cause exponential blow-up of simulation time for proving the security of the larger protocol in the concurrent setting (the same problem encountered in concurrent ZK [26]). This problem was got around in [31] by working in the conditional simulatable input model and sequentially running non-constant number of the underlying argument/proof of knowledge protocols. Concurrent straight-line knowledge-extraction was also investigated in the timing model [42, 43, 41], and in the random oracle model [51]. In general, the issue of concurrent composition of proof of knowledge (POK) could be traced back to the seminal work of Dolev, Dwork and Naor [25].

2 Preliminaries

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

Let $\langle P, V\rangle$ be a probabilistic interactive protocol, then the notation $(y_1, y_2) \leftarrow \langle P(x_1), V(x_2)\rangle(x)$ denotes the random process of running interactive protocol $\langle P, V\rangle$ on common input x, where P has private input x_1, V has private input x_2, y_1 is P's output and y_2 is V's output. We assume w.l.o.g. 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.

2.1 Basic Definitions

Definition 2.1 ((public-coin) interactive argument/proof system) *A pair of interactive machines, $\langle P,\ V\rangle$, is called an interactive argument system for a language L if both are probabilistic polynomial-time (PPT) machines and the following conditions hold:*

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

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

Definition 2.2 (statistically/perfectly binding bit commitment scheme) *A pair of PPT interactive machines, $\langle P, V\rangle$, is called a 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*
$\Pr[(\alpha, \beta) \leftarrow \langle P(b), V\rangle(1^n); (t,(t, v)) \leftarrow \langle P(\alpha), V(\beta)\rangle(1^n) : v = b] = 1$.

Computationally hiding. *For all sufficiently large n's, any PPT adversary V^*, the following two probability distributions are computationally indistinguishable:* $[(\alpha, \beta) \leftarrow \langle P(0), V^*\rangle(1^n) : \beta]$ *and* $[(\alpha', \beta') \leftarrow \langle P(1), V^*\rangle(1^n) : \beta']$.

Perfectly Binding. *For all sufficiently large n's, and* any *adversary P^*, the following probability is negligible (or equals 0 for perfectly-binding commitments):* $\Pr[(\alpha, \beta) \leftarrow \langle P^*, V\rangle(1^n); (t,(t, v)) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n); (t',(t', v')) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n) : v, v' \in \{0, 1\} \bigwedge v \neq v']$.

*That is, no (*even computational power unbounded*) adversary P^* can decommit the same transcript of the commitment stage both to 0 and 1.*

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 [7, 35]. Loosely speaking, given a OWP f with a hard-core predict b (cf. [32]), on a security parameter n one commits a bit σ by uniformly selecting $x \in \{0, 1\}^n$ and sending $(f(x), b(x) \oplus \sigma)$ as a commitment, while keeping x as the decommitment information.

For practical perfectly-binding commitment scheme, in this work we use the DDH-based ElGamal (non-interactive) commitment scheme [28]. To commit to a value $v \in Z_q$, the committer randomly selects $u, r \in Z_q$, 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 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 Z_p^* (for more details, see [28]). We also note that in [45] 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 [47, 38]. On a security parameter n, let $PRG : \{0,1\}^n \longrightarrow \{0,1\}^{3n}$ be a pseudorandom generator, the Naor's OWF-based two-round public-coin perfectly-binding commitment scheme works as follows: In the first round, the commitment receiver sends a random string $R \in \{0,1\}^{3n}$ to the committer. In the second round, the committer uniformly selects a string $s \in \{0,1\}^n$ at first; then to commit a bit 0 the committer sends $PRG(s)$ as the commitment; to commit a bit 1 the committer sends $PRG(s) \oplus R$ as the commitment. Note that the first-round message of Naor's commitment scheme can be fixed once and for all and, in particular, can be posted as a part of public-key in the public-key model.

Definition 2.3 (trapdoor bit commitment scheme) *A trapdoor bit commitment scheme (TC) is a quintuple of probabilistic polynomial-time (PPT) 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:*
$\Pr[(TCPK, TCSK) \leftarrow \mathrm{TCGen}(1^n); (c,d) \leftarrow \mathrm{TCCom}(1^n, TCPK, b) :$
$\mathrm{TCKeyVer}(1^n, TCPK) = \mathrm{TCVer}(1^n, TCPK, c, b, d) = 1] = 1.$

Computationally Binding. *For all sufficiently large n's and for any PPT adversary A, the following probability is negligible in n:* $\Pr[(TCPK, TCSK) \leftarrow \mathrm{TCGen}(1^n); (c, v_1, v_2, d_1, d_2) \leftarrow \mathrm{A}(1^n, TCPK) :$

$\mathrm{TCVer}(1^n, TCPK, c, v_1, d_1) = \mathrm{TCVer}(1^n, TCPK, c, v_2, d_2) = 1 \bigwedge v_1, v_2 \in \{0,1\} \bigwedge v_1 \neq v_2].$

Perfectly (or computationally) Hiding. *For all sufficiently large n's and any $TCPK$ such that* $\mathrm{TCKeyVer}(1^n, TCPK) = 1$, *the following two probability distributions are identical (or computationally indistinguishable):* $[(c_0, d_0) \leftarrow \mathrm{TCCom}(1^n, TCPK, 0) : c_0]$ *and*
$[(c_1, d_1) \leftarrow \mathrm{TCCom}(1^n, TCPK, 1) : c_1]$.

Perfect (or Computational) Trapdoorness. *For all sufficiently large n's and any* $(TCPK, TCSK) \in \{\mathrm{TCGen}(1^n)\}$, $\exists v_1 \in \{0,1\}$, $\forall v_2 \in \{0,1\}$ *such that the following two probability distributions are identical (or computationally indistinguishable):*
$[(c_1, d_1) \leftarrow \mathrm{TCCom}(1^n, TCPK, v_1); d_2' \leftarrow \mathrm{TCFake}(1^n, TCPK, TCSK, c_1, v_1, d_1, v_2) : (c_1, d_2')]$ *and*
$[(c_2, d_2) \leftarrow \mathrm{TCCom}(1^n, TCPK, v_2) : (c_2, d_2)]$.

Feige-Shamir trapdoor commitments (FSTC) [30]. Based on Blum's protocol for DHC, Feige and Shamir developed a generic (computationally-hiding and computationally-binding) trapdoor commitment scheme [30], under either 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 a security parameter n.

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

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 an q-by-q matrix of commitments so that the $(\pi(i), \pi(j))^{th}$ entry is a commitment to 1 if $(i, j) \in E$, and is a commitment to 0 otherwise; To commit 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 2.4 (witness indistinguishability WI) *Let $\langle P, V\rangle$ be an interactive system for a language $L \in \mathcal{NP}$, and let R_L be the fixed $\mathcal{NP}$ witness relation for L. That is, $x \in L$ if there exists a w such that $(x, w) \in R_L$. We denote by $view^{P(w)}_{V^*(z)}(x)$ a random variable describing the transcript of all messages exchanged between a (possibly malicious) PPT verifier V^* and the honest prover P in an execution of the protocol on common input x, when P has auxiliary input w and V^* has auxiliary input z. We say that $\langle P, V\rangle$ is witness indistinguishable for R_L if for every PPT interactive machine V^*, and every two sequences $W^1 = \{w^1_x\}_{x\in L}$ and $W^2 = \{w^2_x\}_{x\in L}$ for sufficiently long x, so that $(x, w^1_x) \in R_L$ and $(x, w^2_x) \in R_L$, the following two probability distributions are computationally indistinguishable by any non-uniform polynomial-time algorithm: $\{x, view^{P(w^1_x)}_{V^*(z)}(x)\}_{x\in L,\, z\in\{0,1\}^*}$ and $\{x, view^{P(w^2_x)}_{V^*(z)}(x)\}_{x\in L,\, z\in\{0,1\}^*}$. Namely, for every non-uniform polynomial-time distinguishing algorithm D, every polynomial $p(\cdot)$, all sufficiently long $x \in L$, and all $z \in \{0, 1\}^*$, it holds that*

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

Definition 2.5 (system for argument/proof of knowledge [32, 6]) *Let R be a binary relation and $\kappa : N \to [0, 1]$. We say that a probabilistic polynomial-time (PPT) interactive machine V is a knowledge verifier for the relation R with knowledge error κ if the following two conditions hold:*

- *Non-triviality: There exists an interactive machine P such that for every $(x, w) \in R$ all possible interactions of V with P on common input x and auxiliary input w are accepting.*
- *Validity (with error κ): There exists a polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every interactive machine P^*, every $x \in L_R$, and every $w, r \in \{0, 1\}^*$, machine K satisfies the following condition:*

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

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

  The oracle machine K is called a knowledge extractor.

An interactive 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.

We mention that Blum's protocol for directed Hamiltonian Cycle DHC [8] is just a 3-round public-coin WIPOK for $\mathcal{NP}$, which is recalled below.

Blum's protocol for DHC [8]. The n-parallel repetitions of Blum's basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [8] is just a 3-round public-coin WIPOK for $\mathcal{NP}$ (with knowledge error 2^{-n}) under any one-way permutation (as the first round of it involves one-round perfectly-binding commitments of a random permutation of G). But it can be easily modified into a 4-round public-coin WIPOK for $\mathcal{NP}$ under any OWF by employing Naor's two-round (public-coin) perfectly-binding commitment scheme [47]. The following is the description of Blum's *basic* protocol for DHC:

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

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

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

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

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

We remark that the WI property of Blum's protocol for HC relies on the hiding property of the underlying perfectly-binding commitment scheme used in its first-round.

2.2 Σ and Σ_{OR} Protocols

Definition 2.6 (Σ-protocol [13]) *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 following 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, z stand for the first, the second and the third message respectively and e is assumed to be a string of length k (such that 1^k is polynomially related to the security parameter 1^n) selected uniformly at random in $\{0, 1\}^k$.*
- *Special honest verifier zero-knowledge (SHVZK). There exists a probabilistic polynomial-time (PPT) 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.*

A Σ-protocol is called *partial witness-independent*, if the generation of its first-round message is independent of (i.e., without using) the witness for $x \in L$. We remark that most Σ-protocols in the literature satisfy this property. In particular, Blum's 3-round public-coin WIPOK for DHC [8] is just a partial witness-independent Σ-protocol for $\mathcal{NP}$. For a good survey of Σ-protocols and their applications, the reader is referred to [17].

Σ-Protocol for DLP [54]. The following is a Σ-protocol $\langle P, V\rangle$ proposed by Schnorr [54] for proving the knowledge of discrete logarithm, w, for a common input of the form (p, q, g, h) such that

$h = g^w \ mod \ p$, where on a security parameter n, p is a uniformly selected n-bit prime such that $q = (p-1)/2$ is also a prime, g is an element in Z_p^* of order q. It is also actually the first efficient Σ-protocol proposed in the literature.

- P chooses r at random in Z_q and sends $a = g^r \ mod \ p$ to V.
- V chooses a challenge e at random in Z_{2^k} and sends it to P. Here, k is fixed such that $2^k < q$.
- P sends $z = r + ew \ mod \ q$ to V, who checks that $g^z = ah^e \ mod \ p$, that p, q are prime and that g, h have order q, and accepts iff this is the case.

The OR-proof of Σ-protocols [14]. One basic construction with Σ-protocols allows a prover to show that given two inputs x_0, x_1, it knows a w such that either $(x_0, w) \in R_0$ or $(x_1, w) \in R_1$, but without revealing which is the case. Specifically, given two Σ-protocols $\langle P_b, V_b \rangle$ for R_b, $b \in \{0, 1\}$, with random challenges of, without loss of generality, the same length k, consider the following protocol $\langle P, V \rangle$, which we call Σ_{OR}. The common input of $\langle P, V \rangle$ is (x_0, x_1) and P has a private input w such that $(x_b, w) \in R_b$.

- P computes the first message a_b in $\langle P_b, V_b \rangle$, using x_b, w as private inputs. P chooses e_{1-b} at random, runs the SHVZK simulator of $\langle P_{1-b}, V_{1-b} \rangle$ on input (x_{1-b}, e_{1-b}), and 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 s and sends it to P.
- P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b to challenge e_b using (x_b, a_b, e_b, w) as input. He sends (e_0, z_0, e_1, z_1) to V.
- V checks that $s = e_0 \oplus e_1$ and that conversations (a_0, e_0, z_o), (a_1, e_1, z_1) are accepting conversations with respect to inputs x_0, x_1, respectively.

Theorem 2.1 [14] *The protocol Σ_{OR} above is a Σ-protocol for R_{OR}, where $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R_0$ or $(x_1, w) \in R_1\}$. Moreover, Σ_{OR}-protocols are witness indistinguishable (WI) argument/proof of knowledge systems.*

The SHVZK simulator of Σ_{OR} [14]. For a Σ_{OR}-protocol of the above form, denote by S_{OR} the perfect SHVZK simulator of it and denote by S_b the perfect 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 as follows: It firstly 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 $((\hat{a}_0, \hat{a}_1), \hat{e}, (\hat{e}_0, \hat{z}_0, \hat{e}_1, \hat{z}_1))$.

2.3 Concurrent soundness and concurrent zero-knowledge in the BPK Model

We recall the definitions of concurrent soundness and concurrent zero-knowledge in the BPK model (cf. [10, 46, 23, 50]).

Honest players in the BPK model. The BPK model 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.
- $P(1^n, x, w, F, id, \gamma)$, an honest prover that is a polynomial-time interactive machine, where 1^n is a security parameter, x is a $poly(n)$-bit string in L, w is an auxiliary input, F is a public-file, id is a verifier identity, and γ is its random-tape.
- V, an honest verifier that is a polynomial-time interactive machine working in two stages.

1. Key generation stage. V, on a security parameter 1^n and a random-tape r, outputs a key pair (PK, SK). V then registers PK in F as its public-key while keeping the corresponding secret key SK in secret.
2. Proof stage. V, on inputs SK, $x \in \{0,1\}^{poly(n)}$ and a random tape ρ, performs an interactive protocol with a prover and outputs "accept x" or "reject x".

The malicious concurrent prover and concurrent soundness in the BPK model. Let V be an honest verifier with public-key PK and secret-key SK, where (PK, SK) is the output of the key generation stage of V on a security parameter 1^n and a random string r. An s-concurrent malicious prover P^* in the BPK model, for a positive polynomial s, is a probabilistic polynomial-time Turing machine that, on a security parameter 1^n and an auxiliary string $z \in \{0,1\}^*$ (e.g., information collected from protocol executions w.r.t. other public-keys that are generated independently of the public-key PK at hand), performs an s-concurrent attack against V as follows (in two stages):

In the first stage, on inputs $(1^n, PK, z)$ P^* first generates an auxiliary string $\tau \in \{0,1\}^*$ (that, in particular, could include z and some side information about the secret-key SK). Then, in the second stage (i.e., proof stage), with the auxiliary information τ, P^* can perform concurrently at most $s(n)$ interactive protocols (sessions) with (the proof stage of) V as follows: If P^* is already running $i-1$ $(1 \le i \le s(n))$ 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 $1 \le j < i$) and initiate a new session with the proof stage of $V(SK, x_i, \rho_i)$; P^* can output a message for any running protocol, and always receive promptly the response from V (that is, P^* controls at its wish the schedule of the messages being exchanged in all the concurrent sessions). We stress that in different sessions V uses independent random-tapes in its proof stage (that is, $\rho_1, \cdots, \rho_{s(n)}$ are independent random strings). Note that extension to the general case, where P^* interacts with instances of multiple verifiers with multiple (independently generated) public-keys, is direct.

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

The malicious 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 τ, works in two stages:

Stage-1 (key-generation stage). V^* receives $s(n)$ strings $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ of length $poly(n)$ each, where x_i might be equal to x_j, $1 \le i, j \le s(n)$, and outputs an arbitrary public-file F and a list of (without loss of generality) $s(n)$ identities $id_1, \cdots, id_{s(n)}$.

Stage-2 (proof stage). Starting from the final configuration of Stage-1, V^* concurrently interacts with $s(n)^2$ instances of the honest prover P: $P(x_i, w_i, id_j, \gamma_{(i,j)})$, where $1 \le i, j \le s(n)$, $(x_i, w_i) \in R_L$ and $\gamma_{(i,j)}$'s are independent random strings. In this stage, V^* controls at its wish the schedule of the messages being exchanged in all the concurrent sessions. In particular, P^* can output a message for any running session dynamically based on the transcript up to now, and always receive promptly the response from P. V^* finally outputs its "view" of the interactions, i.e., its random tape and messages received from all the $s(n)^2$ prover instances. We denote by $view_{V^*(\tau)}^{\{P(x_i,w_i,id_j,\gamma_{(i,j)})'s\}}(\bar{\mathbf{x}})$ the random variable describing the view of V^* in this experiment.

Definition 2.7 (concurrent zero-knowledge in the BPK model) *A protocol $\langle P, V\rangle$ is (black-box) concurrent zero-knowledge for a language $L \in \mathcal{NP}$ 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: $\{view_{V^*(\tau)}^{\{P(x_i,w_i,id_j,\gamma_{(i,j)})'s\}}(\bar{\boldsymbol{x}})\}_{\bar{\boldsymbol{x}} \in L^{s(n)}, \tau \in \{0,1\}^*}$ and $\{S(\bar{\boldsymbol{x}}, \tau)\}_{\bar{\boldsymbol{x}} \in L^{s(n)}, \tau \in \{0,1\}^*}$.*

3 Concurrent Knowledge-Extraction: Motivation and Formulation

In this section, we first present a concurrent interleaving and malleating attack on the concurrent ZK protocol of [23] that is both *concurrently sound* and *normal argument of knowledge* in the BPK model, showing that concurrent soundness and normal arguments of knowledge do not guarantee concurrent verifier security in the public-key model. This attack serves a good motivation for understanding "possession of knowledge on the Internet," i.e., the subtleties of concurrent knowledge-extraction in the public-key model. We then formulate concurrent knowledge-extraction, in light of the concurrent interleaving and malleating attack and in the spirit of non-malleability formulation of [25], and show that it is strictly stronger than concurrent soundness in the public-key model under any OWF.

3.1 The concurrent interleaving and malleating attack

Let us first recall the protocol structure of the CZK protocol of [23].

Key-generation. Let f_V be a OWF that admits Σ-protocols. On a security parameter n, each verifier V randomly selects two elements in the domain of f_V, x_V^0 and x_V^1 of length n each, computes $y_V^0 = f_V(x_V^0)$ and $y_V^1 = f_V(x_V^1)$. V publishes (y_V^0, y_V^1) as its public-key while keeping x_V^b as its secret-key for a randomly chosen b from $\{0,1\}$. (For OWF-based implementation, V also publishes a random string r_V of length $3n$ that serves the first-round message of Naor's OWF-based perfectly-binding commitment scheme [47].)

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_V^0 or y_V^1, by executing the Σ_{OR}-protocol on (y_V^0, y_V^1) in which V plays the role of knowledge prover. It is additionally required that the first-round message of the Σ_{OR}-protocol is generated without using the preimage of either y_V^0 or y_V^1 (i.e., *partial witness-independent*). Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol of this phase respectively. Here e_V is the random challenge sent by the prover to the verifier. (For OWF-based implementation, P sends a random string r_P of length $3n$ on the top, which serves 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 goto Phase-2. Otherwise, P aborts.

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

Phase-3. Phase-3 runs essentially the underlying Σ-protocol for L but with the random challenge 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 computed also 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 incomplete CZK protocol of [59] (Figure-3, page 17), and can be implemented based on any OWF. The key difference in the actual implementations of [59, 23] is that [23] uses a special trapdoor commitment scheme in Phase-2, where the decommitment formation to 0 or 1 is in turn committed in two statistically-binding commitments. This technique is critical for achieving concurrent soundness, the reader is referred to [23] for more details. We remark that the differences in actual implementations do not invalidate the following attack, which is presented w.r.t. a more general protocol structure.

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

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

1. P^* initiates the first 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 by a'_V, of the Σ_{OR}-protocol of Phase-1 of the first session on common input (y_V^0, y_V^1) (i.e., V's public-key), P^* suspends the first session.

2. P^* initiates a second session with V, and works just as the honest prover does in Phase-1 and Phase-2 of the second session. We denote by $c_{\hat{e}}$ the Phase-2 message of the second session (i.e., $c_{\hat{e}}$ commits to a random string $\hat{e}$ of length n). When P^* moves into Phase-3 of the second session and needs to send V the first-round message, denoted by a_P, of the Σ-protocol of Phase-3 of the second session *on common input* $(\hat{x}, y_V^0, y_V^1)$, 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 by $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$".
    - P^* sets $a_P = (a_{\hat{x}}, a'_V)$ and sends a_P to V as the first-round message of the Σ-protocol of Phase-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 Phase-3 of the second session, denoted by q (i.e., the random challenge from V), P^* sets $e_P = \hat{e} \oplus q$ and then suspends the second session.

3. P^* continues the first session, and sends $e'_V = \hat{e} \oplus q \oplus e_{\hat{x}} = e_P \oplus e_{\hat{x}}$ as the second-round message of the Σ_{OR}-protocol of Phase-1 of the first session.

4. After receiving the third-round message of the Σ_{OR}-protocol of Phase-1 of the first session, denoted by z'_V, P^* suspends the first session again.

5. P^* continues the execution of the second session again, reveals $\hat{e}$ committed in Phase-2 of the second session, and sends to V $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V))$ and the decommitment information of $\hat{e}$ 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_V^0 or

y_V^1, and furthermore $e_{\hat{x}} \oplus e'_V = e_P = \hat{e} \oplus q$. According to the description of Σ_{OR} (presented in Section 2), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation of Phase-3 of the second-session on common input $(\hat{x}, y_V^0, y_V^1)$. That is, P^* successfully convinced V of the statement "$(\hat{x}, (y_V^0, y_V^1)) \in L$" (*even for* $\hat{x} \notin \hat{L}$) in the second session *but without knowing any corresponding $\mathcal{NP}$-witness*! We remark that mixing the public key structure as part of the language is a natural attack strategy for the public-key model (a different demonstration of this was given in [57]).

3.2 Formulating concurrent knowledge-extraction in the public-key model

Now, we proceed to formulate concurrent verifier security in light of the above attack. Note that the above attack is of man-in-the-middle nature, and is related to malleability of protocols. The informal intuition for concurrent verifier security is: for any x if P^* can convince V (with public-key PK) of "$x \in L$" by concurrent interleaving interactions, then it must "know" a witness for the statement. More formally, for any x if P^* can convince V (with public-key PK) of "$x \in L$" 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. But, such a definition does *not* work in the public-key model. For example, the language may be related to PK, and thus the extracted witness may be related to SK (actually, for the malicious prover strategy of the above attack, the extracted witness will just be the same secret-key used by the knowledge-extractor). But, in knowledge-extraction the PPT extractor may have already possessed SK. To solve this subtlety, we require that the extracted witness is *independent* of SK. But, the problem here is how to formalize such independence? We solve this in the spirit of non-malleability formulation [25]. That is, we consider the message space (distribution) of SK, and such independence is informally formulated as follows: for any polynomial-time computable relation R and any auxiliary information τ used by P^* in its main proof stages, let SK be the real secret-key and SK' is an element randomly and independently distributed over the space of SK, then we require that the probability $\Pr[R(w, SK, \tau) = 1]$ is negligibly close to $\Pr[R(w, SK', \tau) = 1]$, where w is the witness extracted by the knowledge extractor. This captures the fact that P^* does, in fact, "know" a "witness" for $x \in L$.

Definition 3.1 *[concurrent knowledge-extraction (CKE) in the public-key model] We say that a protocol $\langle P, V \rangle$ for an $\mathcal{NP}$-language L (with $\mathcal{NP}$-relation R_L) is* concurrently knowledge-extractable in the public-key model, *if for any positive polynomial $s(\cdot)$, any s-concurrent malicious prover P^* defined in Section 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 $\{0,1\}^* \cup \{\bot\}$), the following hold:*

Simulatability. *S, on inputs 1^n and z, outputs a pair (str, sta), where str is a simulated transcript that is computationally indistinguishable from the real view of P^*, and sta is some state information to be transferred to E. We denote by (PK, SK) the simulated key-pair generated by S (that emulates the key-generation of the honest verifier), and by τ the output of $P^*(1^n, PK, z)$ in its first stage, where τ is the auxiliary information to be transferred to the proof stages of P^* that could, in particular, include z and some side information about SK.*

Secret-key independent knowledge-extraction. *E, on inputs $(1^n, str, sta)$, outputs witnesses to all statements proved in accepting sessions in str. Specifically, E outputs a list of strings $\overline{w} = (w_1, w_2, \cdots, w_{s(n)})$, satisfying the following:*

- *w_i is set to be $\bot$, if the i-th session in str is not accepting (due to abortion or verifier verification failure), where $1 \le i \le s(n)$.*
- *In any other cases, with overwhelming probability $(x_i, w_i) \in R_L$, where x_i is the common input selected by P^* for the i-th session in str.*
- knowledge-extraction independence (KEI): *For any i, $1 \le i \le s(n)$, and any auxiliary information τ (used by P^* in the proof stages), $\Pr[R(w_i, SK, \tau) = 1]$ is negligibly close to*

$\Pr[R(w_i, SK', \tau) = 1]$, *where SK' is an element randomly and independently distributed over the space of SK.*

The probability is taken over the randomness of S in the key-generation stage (i.e., the randomness for generating (PK, SK)) and in all proof stages, the randomness of E, the choice of SK' and the randomness of P^*.

Some comments about the CKE definition are in place.

We first note that the above CKE formulation follows the simulation-extraction approach of [53] (which is also used in [4]). Here, the key augmentation and difference is that in the public-key model the property of knowledge-extraction independence (KEI) is explicitly required. Note that this requirement does not apply to protocols in the plain model, as the simulator/extractor and honest verifiers do not possess secret values there.

The CKE formulation, i.e., Definition 3.1, captures the following natural intuition for concurrent verifier security in the public-key model: for any x suppose the s-concurrent malicious prover P^* can convince the honest verifier (in real execution w.r.t. real public-key) of the statement "$x \in L$" in one of the $s(n)$ concurrent sessions with non-negligible probability p_x, then it must "know" a witness to $x \in L$. The reasoning is as follows: for any x, suppose P^* can convince the honest verifier (in real execution w.r.t. real public-key) of the statement "$x \in L$" with non-negligible probability p_x, then it must also convince the simulator/extractor (in simulation/extraction w.r.t. a simulated public-key) of the same statement with probability negligibly close to p_x, otherwise the simulatability of Definition 3.1 is violated. Then, the secret-key independent knowledge-extraction property says that, with probability negligibly close to p_x, a witness to $x \in L$ will be extracted that is independent of the secret-key used by the simulator/extractor.

Note that in the KEI formulation, we do not require SK' to be independent of PK. Such a requirement will trivially make the definition meaningless. The reason is that the language and statements being proved by malicious prover may be public-key specific, e.g., a function of the public-key, and thus may be always related to SK (but can be related to an independent and random SK' with negligible probability). This, in particular, implies that verifier's public-key cannot be unique, which is the underlying reason that our solution intrinsically employs the key-pair trick (that is reminiscent of the key-pair encryption [49] and was used for CZK in the BPK model in the incomplete work of [59]).

Extensions and deeper discussions on the CKE formulation are presented in Section 6.

Proposition 3.1 *Assuming any OWF, CKE is strictly stronger than concurrent soundness in the public-key model.*

Proof. (of Proposition 3.1) It's easy to see that CKE implies concurrent soundness in the public-key model. Then the proposition is direct from the attack demonstrated in Section 3.1 on the CZK protocol of [23] 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, suppose $\hat{x} \notin \hat{L}$ or just $\hat{L}$ is empty, the witness extracted by any polynomial-time knowledge-extraction algorithm E (with $SK = x_V^b$ as its input) must be the preimage of either y_V^0 or y_V^1. But, according to the one-wayness of f_V used in the key-generation stage, with overwhelming probability the extracted witness will be the preimage of y_V^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_V^{1-b} is received externally as its input.) Define the relation R as: $R(w, SK, \cdot) = 1$ if $f_V(w) = f_V(SK)$. Then, conditioned on E outputs a witness, the extracted witness (i.e., the preimage of y_V^b) is always related to $SK = x_V^b$, but can be related to a random and independent SK' with negligible probability. Thus, the CZK protocol of [23] is not concurrently knowledge-extractable in the public-key model. □

The general protocol $\langle P, V\rangle$

Key Generation. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be any OWF, where 1^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 private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^{poly(n)}$ for $x \in L$. Here, we assume w.l.o.g. that the witness for any $x \in L \cap \{0,1\}^{poly(n)}$ is of the same length $poly(n)$ (in particular, consider the $\mathcal{NP}$-complete language DHC).

Stage-1. V proves to P that it knows a preimage to one of y_0, y_1, by running a *partial witness independent* 3-round public-coin WI proof of knowledge (WIPOK) 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 $c_w = C(w, r_w)$ 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.

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 a 3-round public-coin WIPOK protocol for $\mathcal{NP}$. The witness used by P is (w, r_w).

Figure-1. The general CZK-CKE argument for $\mathcal{NP}$ in the BPK model.

4 General CZK-CKE for $\mathcal{NP}$ in the BPK Model

In this section, we present the general construction for CZK-CKE arguments for $\mathcal{NP}$ in the BPK model.

The underlying principles behind our solution are the following: First, for any statement to be proved by the prover, we require that it first commits to a witness for that statement, and then proves to the verifier that the committed value is indeed a witness for the statement being proved (this paradigm was also previously considered, e.g., in [12, 44]). Secondly, this proof is not done directly. Rather, the proof of witness is combined with a tool reminiscent of the key-pair encryption of [49] (that was also used in [59] for concurrent ZK in the BPK model), where the verifier pair of keys are considered. The prover gives a compound OR-proof to a modified language based on the witness commitments and the verifier's public keys. The jointly generated statement for the compound proof allows both the efficient simulation (for zero-knowledge) and correct and efficient concurrent witness extraction (for proving soundness) in the BPK model.

The general construction is depicted in Figure 1 (page 14).

If we use the OWP-based perfectly-binding commitments [32], the protocol depicted in Figure-1 can be based on any OWP and can be combined into 4 rounds (i.e., round-optimal [46]). The protocol also can be based on any OWF by using Naor's OWF-based statistically-binding commitments [47] (in this case Stage-1 is actually of 4 rounds). In the later case, the combined protocol runs in 5 rounds, and V could also publish a random string as a part of its public-key, serving as the first-round message of Naor's commitment scheme.

On the intuitive necessity of c_w and c_{sk}. We note that the rough structure of the protocol depicted in Figure-1 is similar to that of the (sequentially-sound) CZK argument proposed in [59] (that is in fact is a variant of the Feige-Shamir ZK arguments [30], actually the version presented in [29], in the public-key model). The Feige-Shamir ZK in the public-key model amounts to the variant protocol without both c_w and c_{sk}. We stress that in the context of the above structure, mandating commitments c_w and c_{sk} of Stage-2 plays a very crucial role for achieving concurrent knowledge-extraction in the public-key model. Specifically, for protocol variants without either c_w or c_{sk}, concrete attacks exist, showing that they are not concurrently knowledge-extractable. In particular, it shows that the protocol of [19], which is the fixed protocol of [59] amounting to variant protocol without c_w, is not concurrently knowledge-extractable. Details are presented in Appendix B.

For presentation simplicity, we provide security analysis w.r.t. the OWP-based construction. The extension to the OWF-based case is direct.

Theorem 4.1 *Under any OWP, there is a round-optimal concurrently knowledge-extractable concurrent ZK argument for $\mathcal{NP}$ in the BPK model.*

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

Concurrent zero-knowledge.

We first consider a mental simulator M that takes as input all secret-keys corresponding to all public-keys registered in the public-key file, *in case the corresponding secret-keys exist.*

For any $s(n)$-concurrent malicious verifier V^*, M runs V^* as a subroutine on input $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ (where x_i might equal x_j, $1 \leq i, j \leq s(n)$ and $i \neq j$), the public file $F = \{PK_1, \cdots, PK_{s(n)}\}$ and all assumed existing secret-keys. M works just as the honest prover does in Stage-1 of any session. In Stage-2 of any session on a common input x_i and with respect to a public-key PK_j (i.e, the i-th session w.r.t PK_j, $1 \leq i, j \leq s(n)$), M computes $c_w^{(i)} = C(0^{poly(n)}, r_w^{(i)})$ and $c_{sk}^{(i)} = C(SK_j, r_{sk}^{(i)})$, where SK_j is the secret-key corresponding to PK_j for which we assume it exists and M knows. Then, M runs the WIPOK protocol with V^* in Stage-3 of the session with $(SK_j, r_{sk}^{(i)})$ as its witness.

To show the output of M is indistinguishable from the view of V^* in real concurrent interactions, we consider another mental simulator M'. M' takes both the witnesses for $\bar{\mathbf{x}} = \{x_1, \cdots, x_{s(n)}\}$ and all the secret-keys corresponding to public-keys registered in F (in case the corresponding secret-keys exist). M' works just as M does, but with the following exception: for any i, j, $1 \leq i, j \leq s(n)$, in Stage-2 of the i-th session on common input x_i w.r.t PK_j, M' computes $c_w = C(w_i, r_w^{(i)})$, where w_i is the witness for the common input x_i. Note that the witness used by M' in Stage-3 is still SK_j, just as M does. That the output of M' is indistinguishable from that of M is from the computational hiding property of the perfectly-binding commitment scheme C used in Stage-2. Otherwise, by a simple hybrid argument, we can violate the hiding property of the underlying commitment scheme C.

We now consider another mental simulator M'' that mimics M' with the following exception: for any i, j, $1 \leq i, j \leq s(n)$, in Stage-3 of the i-th session on common input x_i w.r.t PK_j, the witness used by M'' is w_i, rather than SK_j as used by M'. By hybrid arguments, the output of M'' is indistinguishable from that of M' by the WI property of underlying WIPOK protocol of Stage-3. Also, by hybrid arguments, the output of M'' is also indistinguishable from the view of V^* in real concurrent interactions by the computational hiding property of the underlying perfectly-binding commitment scheme C used in Stage-2.

This establishes that the output of M is indistinguishable from the view of V^* in real concurrent interactions. To build a PPT simulator S from scratch, where S does not know any secret-keys corresponding to public-keys in the public file, we resort to the technique developed in [10]. Specifically, S works in $s(n) + 1$ phases. In each phase, S either successfully finishes the simulation, or "covers" a new public-key for which it has not known the corresponding secret-key up to now in case V^* successfully finishes the Stage-1 interactions w.r.t. that public-key. Key covering is guaranteed by the POK property of Stage-1 interactions. For more details, see [10].

Concurrent knowledge-extraction.

According to the CKE formulation, we need to build two algorithms (S, E). The simulator S, on inputs $(1^n, z)$, 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 $(1^n, PK, z)$ to get the side information τ to be used by P^* in the proof stages. 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', τ) and the random coins used by S. Note that the simulated transcript is identical to that of P^* in real execution.

The knowledge-extraction process is similar to that of [53]. Note that we need to extract 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 i-th session. E_i emulates S with the *fixed* random coins included in sta, with the exception that the random challenge (i.e., the second-round message) of the WIPOK of Phase-3 in the i-th session is no longer emulated internally, but received externally. Note that the views of P^* in the experiments S and E_i are still identical. Furthermore, the experiment E_i amounts to the execution of the WIPOK of Phase-3 between a stand-alone prover and an honest verifier. Suppose the i-th session w.r.t. common input x_i is accepting (note that otherwise we do not need to extract a witness and the witness is set to be "$\perp$"), by applying the stand-alone knowledge-extractor on E_i, we can extract a witness w_i satisfying one of the following (note that w_i is determined by the perfectly-binding commitments of Stage-2):

Case-1. $\sigma = 1 - b$ and $y_\sigma = f(w_i)$, where b is the random bit chosen by the honest verifier in the key-generation stage.

Case-2. $\sigma = b$ and $y_\sigma = f(w_i)$.

Case-3. $(x_i, w_i) \in R_L$.

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.

The subtle point here is: by applying the stand-alone knowledge-extractor on E_i, the Phase-1 interactions given by the simulator/extractor would also be rewound, which could reveal the secret-key SK. By hybrid arguments and a deepened investigation (similar to that of [23]), we can get:

Lemma 4.1 *Case-2 occurs with negligible probability.*

Proof (of Lemma 4.1). We consider two experiments: $\mathcal{E}_0$ and $\mathcal{E}_1$. For each $\mu \in \{0,1\}$, $\mathcal{E}_\mu$ mimics the experiment E_i, with the exception that $\mathcal{E}_\mu$ uses fresh random coins and uses s_μ as its witness in Phase-1 interactions for internal emulation of the proof stages (note that (s_0, s_1) is included in sta). Suppose Case-2 occurs with non-negligible probability, then there must exist a bit μ such that applying the (stand-alone) knowledge-extractor on $\mathcal{E}_\mu$ will output the preimage of y_μ with non-negligible probability. Otherwise, Case-2 will trivially occur with negligible probability. Without loss of generality, we assume $\mu = 0$. That is, the knowledge-extractor on $\mathcal{E}_0$ outputs the preimage of y_0 with non-negligible probability (and outputs the preimage of y_1 with negligible probability due to the one-wayness of f). Now we consider the output of the knowledge-extractor on $\mathcal{E}_1$: first, it outputs the preimage of y_0 also with negligible probability; thus, with non-negligible probability the knowledge-extractor on $\mathcal{E}_1$ outputs either the preimage of y_1 or the witness for $x \in L$. Note that here we cannot directly conclude that the knowledge-extractor on $\mathcal{E}_1$ will certainly output the preimage of y_1 with non-negligible probability, as we cannot rely on the assumption that $x \notin L$ (as is done in [23]). Then, we show how to contradict the WI or partial witness independence properties of the POK protocol of Stage-1.

We define a series of hybrid mental experiments $H_1, \cdots, H_{s(n)}$ as follows: for any k, $1 \leq k \leq s(n)$, H_k mimics the behavior of $\mathcal{E}_0$ but with the following exceptions: In Stage-1 of the first k sessions H_k uses s_1 as its witness; and in Stage-1 of the rest $s(n) - k$ sessions it uses s_0 as the witness. Note that H_0 equals the experiment $\mathcal{E}_0$, and $H_{s(n)}$ equals the experiment $\mathcal{E}_1$. As we assume that the (stand-alone) knowledge-extractor on $H_0(= \mathcal{E}_0)$ will output the preimage of y_0 with non-negligible probability (but output the preimage of y_1 with negligible probability), and that $H_{s(n)}(= \mathcal{E}_1)$ will output either a preimage of y_1 or a witness for $x \in L$ with non-negligible probability (but output the preimage of y_0 only with negligible probability). By hybrid arguments, we conclude that there must exist a k, $1 \leq k \leq s(n)$, such that the knowledge-extractor on H_{k-1} outputs the preimage of y_0 with non-negligible probability and the knowledge-extractor on H_k outputs the preimage of y_0 with negligible probability. Then we show how to break the WI property or the partial witness-independent property of the POK protocol of Stage-1, by considering another experiment B.

B mimics H_k with the following exceptions: The Stage-1 interactions of the k-th session are no longer emulated internally, but interacting externally with an external knowledge-prover $\hat{P}_k$ who uses s_δ as the witness for a random bit δ. Note that, if $\hat{P}_k$ uses s_1 as its witness, then the experiment B is identical to H_k, and if $\hat{P}_k$ uses s_0 as its witness, then B is identical to H_{k-1}. Now, we consider two cases:

Case-2.1. The external interactions with $\hat{P}_k$ have finished before the sending of the random challenge (i.e., the second-round message) of Stage-3 of the i-th session.

Case-2.2. The external interactions with $\hat{P}_k$ have not finished on the sending of the random challenge of Stage-3 of the i-th session. This means that $\hat{P}_k$ sends *at most* the first-round message (it may be the case that $\hat{P}_k$ is initiated after the point). Note that the concurrent interleaving and malleating attack described in Section 3.1 is just a demonstration of this case.

If Case-2.1 occurs, then we break the WI property of the WIPOK protocol of Stage-1 as follows: Note that in this case, applying the stand-alone knowledge-extractor on B does not incur rewinding the interactions with $\hat{P}_k$. We can combine the stand-alone knowledge-extractor and the internal emulation of B into a stand-alone (expected polynomial-time) knowledge-verifier interacting with $\hat{P}_k$. If the knowledge-extractor outputs the preimage of y_0, then we also output 0; in any other case, we output a random bit. According to the above hybrid arguments, if $\hat{P}_k$ uses s_0 as its witness, then we will output 0 with probability that is non-negligibly bigger than 1/2; on the other hand, if $\hat{P}_k$ uses s_1 as its witness, then we will output 0 with probability negligibly close to 1/2. Furthermore, using Markov's inequality standard technique shows that (as is done in [51, 56]), if the WI property holds w.r.t. any strict polynomial-time algorithm it also holds with any expected polynomial-time algorithm. This contradicts the WI property of the underlying protocol.

If Case-2.2 occurs, note that B sees *at most* the first-round message of the underlying partial witness-independent WIPOK protocol of Stage-1 of the k-th session at the point of sending the random challenge of Stage-3 in the i-th session. As the first-round message is generated *independently* of the witness used by $\hat{P}_k$, and the witness used by $\hat{P}_k$ is s_δ for a random bit $\delta \in \{0, 1\}$, this means that at the point of sending the random challenge of Stage-3 in the i-th session *for the first time* (here we do not need to consider rewinding), even a *computational power unbounded* algorithm can guess the random bit δ with probability just 1/2. Now, we break this assumption as follows. When B is required to send the second-round message of Stage-3 in the i-th session for the first time, we stop the experiment, and do the following: we open the perfectly-binding commitment $c_{sk}^{(i)}$ (sent by P^* in Stage-2 of the i-th session) by brute force in exponential-time. If we get $f^{-1}(y_0)$ then we output 0, and in any other case output a random bit. Again, by the assumption on H_{k-1} and H_k, we can guess the random bit δ with probability non-negligibly greater than 1/2. This finishes the proof of Lemma 4.1. □

By removing Case-1 and Case-2, we conclude now that for any i, $1 \leq i \leq s(n)$, if the i-th session in *str* is accepting w.r.t. common input x_i selected by P^*, then the knowledge-extractor E will output a witness for $x_i \in L$ (*that is uniquely determined by* $c_w^{(i)}$). To finish the proof, we need to further show that knowledge-extraction is independent of the secret-key used by E (or, S). As $SK' = s_{1-b}$ is randomly and independently distributed over the space of $SK = S_b$, it is enough to establish the following lemma.

Lemma 4.2 *For any auxiliary information τ used by P^* in proof stages, for any polynomial-time computable relation R and any i, $1 \leq i \leq s(n)$, it holds that* $\Pr[R(w_i, SK, \tau) = 1]$ *is negligibly close to* $\Pr[R(w_i, SK', \tau) = 1]$.

Proof (of Lemma 4.2). The proof of Lemma 4.2 is similar to that of Lemma 4.1.

For any i, $1 \leq i \leq s(n)$, we consider the two experiments $\mathcal{E}_0$ and $\mathcal{E}_1$ presented at the beginning of the proof of Lemma 4.1. Denote by e_μ the output of the stand-alone knowledge-extractor on $\mathcal{E}_\mu$, $\mu \in \{0, 1\}$. Note that $\Pr[R(e_b, s_b, \tau) = 1] = \Pr[R(w_i, SK, \tau) = 1]$, as the probabilities are taken over identical distributions. Similarly, $\Pr[R(e_b, s_{1-b}, \tau) = 1] = \Pr[R(w_i, SK', \tau) = 1]$.

We first show that $\Pr[R(e_0, s_1, \tau)) = 1]$ is negligibly close to $\Pr[R(e_1, s_1, \tau) = 1]$, where $SK = s_b$ and $SK' = s_{1-b}$. Otherwise, by using similar hybrid arguments and proof procedure as in the proof of Lemma 4.1, we can break either the WI property or the partial witness independence property of the underlying protocol of Stage-1. We stress that the perfectly-binding commitment c_w of Stage-2 (that *uniquely* determines the witness for $x \in L$) plays a crucial role in the proof, specifically, for contradicting the partial witness independence property of Stage-1 protocol. Actually, as shown in Appendix B, protocol variant without c_w is not concurrently knowledge-extractable in the public-key model (specifically, concrete attacks exist).

Similarly, we get that $\Pr[R(e_0, s_0, \tau)) = 1]$ is negligibly close to $\Pr[R(e_1, s_0, \tau) = 1]$. Thus, we conclude that $\Pr[R(e_b, s_b, \tau) = 1] = \Pr[R(w_i, SK, \tau) = 1]$ is negligibly close to $\Pr[R(e_{1-b}, s_b, \tau) = 1]$. By defining $\upsilon = 1 - b$, we get $\Pr[R(e_{1-b}, s_b, \tau) = 1] = \Pr[R(e_\upsilon, s_{1-\upsilon}, \tau) = 1] = \Pr[R(e_b, s_{1-b}, \tau) = 1]$ that is just identical to $\Pr[R(w_i, SK', \tau) = 1]$, where both b and υ are random bits. □

□

5 Practical Implementation

In the practical implementation of CZK-CKE arguments, the verifier uses the DLP OWF in key-generation stage: $f_{p,q,g}(x) = g^x \mod p$, where p and q are primes, $p = 2q + 1$ and $|p| = n$, and g is an element of Z_p^* of order q. We also assume the DDH assumption holds on the cyclic group indexed by (p, q, g) (i.e., the sub-group of order q of Z_p^*). The common input is $x \in Z_p^*$ of order q. We remark that, as noted in [22, 23], the parameter (p, g, g), specifying the $f_{p,q,g}$ and the common inputs, is set outside the system.

The partial witness independent WIPOK of Stage-1 is replaced by the Σ_{OR} of Schnorr's basic protocol for DLP [54]. The perfectly-binding commitment scheme of Stage-2 is replaced by the DDH-based ElGamal (non-interactive) commitment scheme [28] (recalled in Appendix 2.1). To commit to a value $v \in Z_q$, the committer randomly selects $u, r \in Z_q$, 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** $y_0 = g^w \mod p$ and $\bar{g}_{sk} = g^r \mod p$ and $\bar{h}_{sk} = g^w h^r \mod p$ **or** $y_1 = g^w \mod p$ and $\bar{g}_{sk} = g^r \mod p$ and $\bar{h}_{sk} = g^w h^r \mod p$.

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. Based on the Σ-protocol for DLP [54], such Σ-protocol is described below.

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 Z_p^*.

Prover's private input: $w, r \in Z_q$ 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 randomly selects $t \in Z_q$, computes $a_0 = g^t \mod p$ and $a_1 = h^t \mod p$, sends (a_0, a_1) to the verifier V.

Round-2: V responds back a random challenge e taken randomly from Z_q.

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

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

We give a brief analysis of the above Σ-protocol:

Special soundness: From two accepting conversations w.r.t. the same Round-1 messages, $\{(a_0, a_1), e, (z_0, z_1)\}$ and $\{(a_0, a_1), e', (z_0', z_1')\}$, we can compute $w = \frac{z_0 - z_0'}{e - e'}$, and $r = \frac{z_1 - z_1'}{e - e'}$.

Special HVZK: The SHVZK simulator S works as follows: on a given random challenge $e \in Z_q$, it randomly selects z_0, z_1 from Z_q, then it sets $a_0 = g^{z_0} x^{-e}$ and $a_1 = g^{z_1} \bar{g}^{-e} = h^{z_1} (\bar{h}/x)^{-e}$.

We remark that, although the above practical implementation is for specific number-theoretic language and is based on the DDH assumption, it is indeed very useful in practical scenarios. Practical CZK arguments in the BPK model were also developed in [59, 55, 20] (with sequential or concurrent soundness), but not for the CKE model.

6 Extensions and Discussions

A natural extended notion of CKE is to require that the KEI property formulated in Definition 3.1 not only holds w.r.t. the individual distribution of each extracted witness w_i but also holds w.r.t. the joint distribution of $\overline{w}$. Specifically, the KEI property is reformulated as: $\Pr[R(\overline{w}, SK, \tau) = 1]$ is negligibly close to $\Pr[R(\overline{w}, SK', \tau) = 1]$, where $\overline{w} = (w_1, \cdots, w_{s(n)})$ is the output of E. We call such property as joint KEI (JKEI), and CKE with JKEI as JCKE.

The extended JCKE notion essentially says that: the concurrent malicious prover P^* not only knows witness to each individual accepting session, but it also knows the joint witnesses to all accepting sessions. Clearly, JCKE implies the normal CKE. But, we do not know whether JCKE is *strictly* stronger than CKE under proper assumptions, and suggest it as an interesting open problem for future investigation.

We show that the generic CZK-CKE construction depicted in Figure-1 in Section 4 can be slightly modified to satisfy the extended JCKE notion. The modification is to replace the partial witness-independent WI protocol of Stage-1 by any (constant-round) *statistical/perfect* WI argument/proof of knowledge system. By requiring the statistical WI property, we can waive the partial witness-independence property.

For provable JCKE property of the modified protocol, we first note that Lemma 4.1 still holds but actually with a simplified security analysis. Specifically, by exploiting the statistical WI property of Stage-1 interaction, in the security analysis we do not need to distinguish whether the k-th session is finished or not before the point of sending the random challenge of Stage-3 in the i-th session: Whenever we reach this point, we just open $c_{sk}^{(i)}$ by brute force to violate the statistical WI property of Stage-1 of the k-th session.

This establishes that the output of E, i.e., $\overline{w}$, is well determined by *str* outputted by S (specifically, the perfectly-binding commitments $c_w^{(i)}$'s) and can be got by brute force in exponential-time from the *str*. Now, suppose the JCKE property does not hold for the modified protocol, then we can modify P^* into an exponential-time adversary A that breaks the statistical WI property of Stage-1 as follows: A externally and concurrently runs the statistical WI protocol of Stage-1 on common input PK, playing the role of knowledge-verifier by running P^* as a subroutine and internally emulating Stage-2 and Stage-3 interactions with P^*. Whenever P^* stops, A extracts the well-determined $\overline{w}$ from the transcript by brute force, and applies the assumed distinguishing relation R to guess which secret-key is used by the external knowledge-prover instances. Details are given in the full version.

Finally, we briefly note two simple ways for achieving statistical WI argument/proof of knowledge systems. The first approach is to use in key-generation stage any OWF that admits Σ-protocols with statistical/perfect SHVZK property. Note that in this case, Σ_{OR} is statistical/perfect WI. We remark that the set of OWFs admitting Σ-protocols of perfect SHVZK property is very large, which includes in particular the popular DLP and RSA, etc. This shows that the practical CZK-CKE argument developed in Section 5 satisfies the extended JCKE property. The second approach is to use constant-round statistical WI argument of knowledge for $\mathcal{NP}$ in Stage-1 (in this case, we still can use any OWF for key generation). Such protocol can be got by replacing the perfectly/statistically binding commitments used in Blum's protocol for DHC by any constant-round statistically-hiding commitment scheme, e.g., the one-round schemes of [18, 40] that are based on any collision-resistant hash function or the constant-round scheme of [32] that are based on claw-free collections (statistically-hiding commitments can also be based on general assumptions [48, 39], but are of non-constant rounds).

Comments: We remark that although CKE is seemingly weaker than JCKE, it is still very useful and captures, as clarified in Section 3.2, some nature intuition of concurrent verifier security in the public-key model. In particular, CKE allows the possibilities of minimal hardness assumption (i.e., any OWF) or optimal rounds (based on any OWP) for more general cryptographic systems. For example, when CZK-CKE protocols are used as a building block in larger systems in the public-key model, for provable security of the larger systems, by hybrid arguments the security of the larger systems could often be reduced to the inability of the adversary in question to convince of an (individual) statement in the underlying CZK-CKE system without knowing the corresponding witness. This is one major reason for separately treating CKE and JCKE in this work.

Another natural extension is to consider concurrent knowledge-extraction for concurrent man-in-the-middle adversaries who concurrently interact with both prover and verifier instances. This research line is outside the scope of the current work. We do not deal with concurrent non-malleability in this work, but we suggest that the clarifications and formulation of CKE in the public-key model presented in this work are of independent value and can serve as a basis for formulating and achieving more complex interactive cryptographic protocols in the public-key model. Also, we remark that CZK-CKE arguments are themselves useful tools and can be used in other applications (they are the concurrent version of the highly useful ZK arguments of knowledge in the public-key model).

Acknowledgments. We are grateful to Giovanni Di Crescenzo, Jonathan Katz, Yehuda Lindell, Pino Persiano and Ivan Visconti for many helpful and valuable discussions and comments. We thank Rafael Pass and Alon Rosen for their very kind clarifications on [52, 53] and valuable comments.

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. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *IEEE Symposium on Foundations of Computer Science*, pages 186-195, 2004.

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

[4] B. Barak, M. Prabhakaran, and A. Sahai. Concurrent Non-Malleable Zero-Knowledge. Cryptology ePrint Archive, Report No. 2006/355. Extended abstract appears in FOCS 2006.

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

[6] M. Bellare and O. Goldreich. On Probabilistic versus Deterministic Provers in the Definition of Proofs Of Knowledge. Electronic Colloquium on Computational Complexity, 13(136), 2006. Available also from Cryptology ePrint Archive, Report No. 2006/359.

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

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

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

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

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

[12] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *ACM Symposium on Theory of Computing*, pages 494-503, 2002.

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

[14] R. Cramer, I. Damgard and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Y. Desmedt (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1994, LNCS 893*, pages 174-187. Springer-Verlag, 1994.

[15] 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.

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

[17] I. Damgard. Lecture Notes on Cryptographic Protocol Theory, BRICS, Aarhus University, 2003.

[18] I. Damgard, T. Pedersen and B. Pfitzmann. On the Existence of Statistically-Hiding Bit Commitment and Fail-Stop Signatures. In Crypto 1993.

[19] Y. Deng and D. Lin. Resettable Zero Knowledge in the Bare Public-Key Model under Standard Assumption. Cryptology ePrint Archive, Report No. 2006/239.

[20] Y. Deng, G. Di Crescenzo, and D. Lin. Concurrently Non-Malleable Zero-Knowledge in the Authenticated Public-Key Model. Cryptology ePrint Archive, Report No. 2006/314, September 12, 2006.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] T. El Gamal. A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31: 469-472, 1985.

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

[30] 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.

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

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

[33] 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.

[34] 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.

[35] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All language in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery*, 38(1): 691-729, 1991. Preliminary version appears in [33, 34].

[36] 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.

[37] 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.

[38] 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.

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

[40] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In Crypto 1996.

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

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

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

[44] J. Kilian. Uses of Randomness in Algorithms and Protocols. MIT Press, Cambridge, MA, 1990.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] R. Ostrovsky, G. Persiano and I. Visconti. Concurrent Non-Malleable Witness Indistinguishability and Its Applications. Electronic Colloquium on Computational Complexity, 13(95), 2006. Available also from Cryptology ePrint Archive, Report No. 2006/256.

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

[52] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *ACM Symposium on Theory of Computing*, pages 533-542, 2005.

[53] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. In *IEEE Symposium on Foundations of Computer Science*, pages 563-572, 2005.

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

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

[56] M. Yung and Y. Zhao. Concurrently Knowledge-Extractable Resettable-ZK in the Bare Public-Key Model. Electronic Colloquium on Computational Complexity, 12(48), 2005.

[57] 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.

[58] Y. Zhao, et al. Personal communications.

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

A Existing CNM Formulations Do Not Capture CKE in the Public-Key Model

This section is taken from [58] for a clear reference.

Recently, we noted some seemingly related but different works [50, 20]. The works of [50, 20] consider concurrent non-malleability (CNM) for ZK arguments in the (slightly stronger) *authenticated* BPK model. But, a careful investigation shows that the CNM formulations in [50, 20] do not capture concurrent knowledge-extraction in the public-key model.

A.1 Observations on the work of [20]

We first observe that the CNM formulation of [20] does not capture concurrent knowledge-extraction in the public-key model. Specifically, although it explicitly requires knowledge-extraction, it does not require the knowledge-extraction independence (KEI) property as formulated here, which is however crucial for concurrent knowledge-extraction in the public-key model. In particular, concrete attacks exist against the protocol of [20], showing that it fails in achieving concurrent knowledge-extraction in the public-key model (actually, more subtleties regarding [20] are observed in [58]).

A.1.1 The attack on the protocol of [20]

Let us first recall the protocol structure of the CZK protocol of [20].

Key-generation. Let (KG_0, Sig_0, Ver_0) and (KE_1, Sig_1, Ver_1) be two signature schemes that secure against adaptive chosen message attacks. On a security parameter 1^n, each verifier V randomly generates two pair $(verk_0, sigk_0)$ and $(verk_1, sigk_1)$, where $verk$ is the signature verification key and $sigk$ is the signing key. V publishes $(verk_0, verk_1)$ as its public-key while keeping $sigk_b$ as its secret-key for a randomly chosen b from $\{0,1\}$ (V discards $sigk_{1-b}$).

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 either $sigk_0$ or $sigk_1$, by executing the (partial witness-independent) Σ_{OR}-protocol on $(verk_0, verk_1)$ in which V plays the role of knowledge prover. Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol of this phase respectively. Here e_V is the random challenge sent by the prover to the verifier.

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

Phase-2. P generates a key pair (sk, vk) for a one-time strong signature scheme. Let COM be a commitment scheme. The prover randomly selects random strings $s, r \in \{0,1\}^{poly(n)}$, and computes $C = COM(s, r)$ (that is, P commits to s using randomness r). Finally, P sends (C, vk) to the verifier V.

Phase-3. By running a Σ_{OR}-protocol, P proves to V that it knows either a witness w for $x \in L$ OR the value committed in C is a signature on the message of vk under either $verk_0$ or $verk_1$. Denote by a_P, e_P, z_P, the first-round, the second-round and the third-round message of the Σ_{OR} of Phase-3. Finally, P computes a one-time strong signature δ on the whole transcript with the signing key sk generated in Phase-2.

Verifier's decision. V accepts if and only if the Σ_{OR}-protocol of Phase-3 is accepting, and δ is a valid signature on the whole transcript under vk.

Note: The actual implementation of the DDL protocol combines rounds of the above protocol. But, it is easy to see that round-combination does not invalidate the following attacks.

The attack.

The following CMIM attack enables A to malleate the interactions of Phase-1 of one session into a successful conversation of another concurrent session for different (but verifier's public-key related) statements without knowing any corresponding $\mathcal{NP}$-witnesses, showing clearly that the protocol of [20] is not concurrently knowledge-extractable in the public-key model.

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 $PK = (verk_0, verk_1)$, we define a new language $L = \{(\hat{x}, verk_0, verk_1) | \exists w \ s.t. \ (\hat{x}, w) \in R_{\hat{L}} \text{ OR } w = sigk_b \text{ for } b \in \{0, 1\}\}$. Note that for any string $\hat{x}$ (whether $\hat{x} \in \hat{L}$ or not), the statement "$(\hat{x}, verk_0, verk_1) \in L$" is always true as $PK = (verk_0, verk_1)$ is honestly generated. Also note that L is a language that admits Σ-protocols (as Σ_{OR}-protocol is itself a Σ-protocol). Now, we describe the concurrent interleaving and malleating attack, in which A successfully convinces the honest verifier of the statement "$(\hat{x}, verk_0, verk_1) \in L$" for *any arbitrary $poly(n)$-bit string $\hat{x}$* (*even when $\hat{x} \notin \hat{L}$*) by concurrently interacting with V (with public-key $(verk_0, verk_1)$) in two sessions as follows.

1. A initiates the first session with V. After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the first session on common input $(verk_0, verk_1)$ (i.e., V's public-key), A suspends the first session.

2. A initiates a second session with V, and works just as the honest prover does in Phase-1 and Phase-2 of the second session. We denote by C, vk the Phase-2 message of the second session, where C is the commitment to a random string and vk is the verification key of the one-time strong signature scheme generated by A (*note that A knows the corresponding signing key sk as (vk, sk) is generated by itself*). When A moves into Phase-3 of the second session and needs to send V the first-round message, denoted by a_P, of the Σ_{OR}-protocol of Phase-3 of the second session *on common input* $(\hat{x}, verk_0, verk_1)$, A does the following:

   - A 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 by $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$".
   - A runs the SHVZK simulator of the Σ-protocol for showing that the value committed in C is a signature on vk under one of $(verk_0, verk_1)$ to get a simulated conversation, denoted by (a_C, e_C, z_C).
   - A sets $a_P = (a_{\hat{x}}, a'_V, a_C)$ and sends a_P to V as the first-round message of the Σ_{OR}-protocol of Phase-3 of the second session, where a'_V is the one received by A in the first session.
   - After receiving the second-round message of Phase-3 of the second session, i.e., the random challenge e_P from V. A suspends the second session.

3. A continues the first session, and sends $e'_V = e_P \oplus e_{\hat{x}} \oplus e_C$ as the second-round message of the Σ_{OR}-protocol of Phase-1 of the first session.

4. After receiving the third-round message of the Σ_{OR}-protocol of Phase-1 of the first session, denoted by z'_V, A suspends the first session again.

5. A continues the execution of the second session again, sends to $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V), (e_C, z_C))$ to V as the third-round message of the Σ_{OR}-protocol of the second session.

6. Finally, A applies sk on the whole transcript *of the second session* to get a (one-time strong) signature δ, and sends δ to V

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 either $sigk_0$ or $sigk_1$, (a_C, e_C, z_C) is an accepting conversation for showing that the value committed in C is a signature on vk under one of $(verk_0, verk_1)$. Furthermore, $e_{\hat{x}} \oplus e'_V \oplus e_C = e_P$, and δ is a valid (one-time strong) signature on the transcript of the second session.This means that, from the viewpoint of V, A successfully convinced V of the statement "$(\hat{x}, verk_0, verk_1) \in L$" in the second session *but without knowing any corresponding $\mathcal{NP}$-witness*!

A.2 Observations on the CNM formulation of [50]

We note that the CNM formulation in the work [50] uses a different indistinguishability-based approach, following that of [52, 53]. Specifically, in the CNM formulation of [50], two experiments are defined (page 19 of [50]): a real experiment w.r.t. a real public-key of an honest verifier (here, denoted PK_V), in which a CMIM adversary mounts CMIM attacks; a simulated experiment run by a simulator/extractor S w.r.t. a simulated public-key (here, denoted PK_S), in which S accesses A and takes a simulated secret-key SK_S. The CNM is then formulated as follows: the distribution of all witnesses used by A in right sessions in the real experiment is indistinguishable from the distribution of the witnesses used by A in right sessions in the simulated experiment. Note that [50] does not require the simulator/extractor to output a simulated indistinguishable transcript.

It appears that the CNM formulation of [50] has already dealt with the issue of knowledge-extraction independence. But, a careful investigation shows that it does not. The reason is as follows:

Firstly, in the real experiment the statements selected by the CMIM adversary A for both left and right sessions can be maliciously related to PK_V (e.g., some function of PK_V), and thus the witnesses extracted for right sessions of the real experiment could be potentially dependent on the secret-key SK_V used by honest players. Note that, as witnessed by the concurrent interleaving and malleating attacks developed in this work, when extracted witnesses are maliciously dependent on SK_V knowledge-extraction does not necessarily capture the intuition that V does know the witnesses to accepting sessions. Similarly, as in the simulated experiment S uses SK_S in simulation/extraction, the witness extracted to right sessions in the simulated experiment could also be maliciously dependent on SK_S. That is, both the witnesses extracted in real experiment and in the simulated experiment may be maliciously dependent on SK_V and SK_S respectively, *but the distributions of them still can be indistinguishable as the distributions of SK_V and SK_S are identical!*.

We note that the above observation only shows that the CNM formulation of [50] does not capture CKE in the public-key model as formulated in this work. Actually, the protocol of [50] seems to be intuitively CKE secure in the public-key model. The key point here is that a protocol proved secure according to the CNM formulation of [50] is not necessarily "concurrently knowledge-extractable" in the public-key model. In summary, we suggest that the clarifications and formulation of CKE in the public-key model presented in this work are of independent value, and has the potential to serve as a basis for achieving more complex interactive cryptographic protocols in the public-key model where arguments about internal states and possession of secrets are crucial.

B Discussion: On the Necessity of c_w and c_{sk}

To show the necessity of the commitments c_w and c_{sk} used in Stage-2 of the protocol depicted in Figure-1, we demonstrate concrete attacks against variants of the protocol without either c_w or c_{sk}, where (partial witness-independent) WIPOK protocols are implemented by Σ_{OR}-protocols.

B.1 The attack against variant protocol without c_w

The variant protocol without c_w, which amounts to the CZK protocol of [19] that is the fixed protocol of [59], is re-depicted in Figure-2 (page 27).

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 OWP, 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 $y_{1-b} = s_{1-b}$. 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 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 a preimage to one of y_0, y_1, by running a *partial witness independent* Σ_{OR}-protocol for $\mathcal{NP}$, e.g., the OR-proof of Blum's protocol for DHC, in which V plays the role of knowledge prover. The witness used by V in this stage is s_b. Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol, respectively.

Stage-2. If V successfully finishes Stage-1, P does the following: it 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 for $\mathcal{NP}$. The witness used by P is w such that $(x, w) \in R_L$. We denote by a_P, e_P, z_P, the first-round, the second-round, and the third-round message of the Σ_{OR}-protocol of this stage, respectively.

Figure-2. Σ_{OR}-based protocol variant without c_w.

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 $L = \{(\hat{x}, y_0, y_1) | \exists w \ s.t. \ (\hat{x}, w) \in R_{\hat{L}} \vee \exists (w, b) \ s.t. \ y_b = f(w) \wedge b \in \{0,1\}\}$. 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 is itself 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 when* $\hat{x} \notin \hat{L}$) 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 by 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 a 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 by 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 by $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, 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 by (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 by 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.

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 OWP, 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 $y_{1-b} = s_{1-b}$. 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 private input. An $\mathcal{NP}$-witness $w \in \{0,1\}^n$ for $x \in L$. Here, we assume w.l.o.g. that the witness for any $x \in L\cap\{0,1\}^n$ is of the same length n (in particular, consider the $\mathcal{NP}$-complete language DHC).

Stage-1. V proves to P that it knows a preimage to one of y_0, y_1, by running a *partial witness independent* Σ_{OR}-protocol for $\mathcal{NP}$, e.g., the OR-proof of Blum's protocol for DHC, in which V plays the role of knowledge prover. The witness used by V in this stage is s_b. Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol, respectively.

Stage-2. If V successfully finishes Stage-1, P does the following: it 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 for $\mathcal{NP}$. The witness used by P is (w, r_w). We denote by a_P, e_P, z_P, the first-round, the second-round, and the third-round message of the Σ_{OR}-protocol of this stage, respectively.

Figure-3. Σ_{OR}-based protocol variant without c_{sk}.

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 by z'_V, P^* suspends the first session again.
5. P^* continues the execution of the second session again, 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 Section 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.*

B.2 The attack against variant protocol without c_{sk}

The variant protocol without c_{sk} is re-depicted in Figure-3 (page 28).

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 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 by 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 a 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 needs to send V the first-round message, denoted by 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 by (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 by 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 by z'_V, P^* suspends the first session again.

5. P^* continues the execution of the second session again, 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} (presented in Section 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.*

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