Resettably-Sound Zero-Knowledge and its Applications (preliminary version)

Boaz Barak Oded Goldreich* Shafi Goldwasser† Yehuda Lindell

Department of Computer Science
Weizmann Institute of Science, Israel

August 7, 2001

Abstract

Resettably-sound proofs and arguments maintain soundness even when the prover can reset the verifier to use the same random coins in repeated executions of the protocol. We show that resettably-sound zero-knowledge arguments for $\mathcal{NP}$ exist if collision-free hash functions exist. In contrast, resettably-sound zero-knowledge proofs are possible only for languages in $\mathcal{P}$/poly.

We present two applications of resettably-sound zero-knowledge arguments. First, we construct resettable zero-knowledge arguments of knowledge for $\mathcal{NP}$, using a natural relaxation of the definition of arguments (and proofs) of knowledge. We note that, under the standard definition of proof of knowledge, it is impossible to obtain resettable zero-knowledge arguments of knowledge for languages outside $\mathcal{BPP}$. Second, we construct a constant-round resettable zero-knowledge argument for $\mathcal{NP}$ in the public-key model, under the assumption that collision-free hash functions exist. This improves upon the sub-exponential hardness assumption required by previous constructions.

We emphasize that our results use non-black-box zero-knowledge simulations. Indeed, we show that some of the results are *impossible* to achieve using black-box simulations. In particular, only languages in $\mathcal{BPP}$ have resettably-sound arguments that are zero-knowledge with respect to black-box simulation.

*Supported by the MINERVA Foundation.
†Supported in part by NTT grant

Contents

1 Introduction

Having gained a reasonable understanding of the security of cryptographic schemes and protocols as stand-alone, cryptographic research is moving towards the study of stronger notions of security. Examples include the effect of executing several instances of the same protocol concurrently (e.g., the malleability of an individual protocol [10]) as well as the effect of executing the protocol concurrently to any other activity (or set of protocols) [6]. Another example of a stronger notion of security, which is of theoretical and practical interest, is the security of protocols under a "resetting" attack. In a resetting attack, a party (being attacked) can be forced to execute a protocol several times while using the same random tape and without the ability to coordinate between these executions (as he may not even be aware of all the executions taking place). The theoretical interest in this notion stems from the fact that randomness plays a pivotal role in cryptographic protocols, and thus the question of whether one needs fresh (and independent) randomness in each invocation of a cryptographic protocol is natural. The practical importance is due to the fact that in many settings it is impossible or too costly to generate fresh randomness on the fly. Moreover, when parties in a cryptographic procotol are implemented by devices which cannot reliably keep state (e.g., smart cards), being maliciously "reset" to a prior state could be a real threat.

1.1 Resettable Provers

Resettability of players in a cryptographic protocol was first addressed by Canetti *et al.* in [7] who considered what happens to the security of zero-knowledge interactive proofs and arguments when the verifier can reset the prover to use the same random tape in multiple executions. Protocols which remain zero-knowledge against such a verifier, are called resettable zero-knowledge (rZK) protocols. Put differently, the question of prover resettability, is whether zero-knowledge is achievable when the prover cannot use fresh randomness in new interactions, but rather is restricted to (re-)using a fixed number of coins.

Resettability implies security under concurrent executions: any rZK protocol constitutes a concurrent zero-knowledge protocol. The opposite direction does not hold, and indeed it was not a-priori clear whether (non-trivial) rZK protocols exist. The main result of Canetti *et al.* answers this question affirmatively, under some standard complexity assumptions. Specifically, assuming the existence of perfectly hiding and computationally binding commitment schemes, resettable zero-knowledge interactive proofs for $\mathcal{NP}$ using polynomially many rounds do exist [7].[1]

In order to obtain a constant-round rZK protocol, Canetti *et al.* introduced a weak public-key model and used a strong intractability assumption - the existence of a perfectly hiding and computationally binding commitment scheme that cannot be broken by sub-exponential size circuits, and not merely by polynomial-size ones. In this model and under that assumption,[2] they were able to construct a constant-round rZK argument system for $\mathcal{NP}$ [7].

On the negative side, [7] point out that resettable zero-knowledge *proofs of knowledge* are impossible to achieve for non-trivial languages, ruling out the use of the Fiat-Shamir [17] paradigm of

[1]The number of rounds was recently improved to poly-logarithmic [26]. Interestingly, poly-logarithmic many rounds are necessary for any protocol that can be shown to be concurrent zero-knowledge via a black-box simulator [8].

[2]An essential use of this sub-exponential hardness assumption, is made when demonstrating the computational-soundness of the protocol. The prover and verifier in the protocol both utilize commitment schemes in an interleaved fashion, which raises the danger of malleability of one commitment scheme with respect to the other. By choosing different security parameters k and K for each commitment scheme such that $2^{k^\epsilon} > 2^K$ and relying on the assumed sub-exponential hardness 2^{t^ϵ} of breaking a commitment scheme with parameter t, successful malleability is ruled out. The same assumption is used by Micali and Reyzin [29] who subsequently achieved resettable zero-knowledge arguments in the public-key model in an optimal number of rounds.

identification protocols based on proofs of knowledge when the provers may be resettable. This impossibility extends to resettable zero-knowledge arguments of knowledge and to resettable witness indistinguishable proofs and arguments of knowledge. All these negative results are with respect to the standard definition of proofs of knowledge (cf. [3]), in which the extractor of knowledge is limited to oracle access to the prover (for detailed discussion see Section 1.3.2).

1.2 Resettable Verifiers

In a similar fashion, one may consider what happens to the soundness of (zero-knowledge) interactive proofs and arguments when the prover can reset the verifier to use the same random tape in multiple concurrent executions.

Informally, we say that an interactive proof or argument achieves *resettable soundness* if a prover cannot convince a verifier of an incorrect statement with non-negligible probability, even when the prover can reset the verifier to use the same random tape in multiple concurrent executions. The verifier resettability question can be recast as whether soundness can be achieved, when the verifier is restricted to (re-)using a fixed number of coins rather than using fresh coins in every interaction.

Resettable-soundness *in the public key model* was already defined and studied by Micali and Reyzin [30]. They showed that the existing rZK protocols in the public-key model (i.e., [7, 29]) are not resettably-sound (i.e., do not maintain soundness when the verifier can be reset). Furthermore, they demonstrated the non-robustness of soundness *in the public key model* by considering several natural notions of soundness (i.e., one-time soundness, sequential soundness, concurrent soundness, and resettable soundness), and showing separations between these notions.

1.3 Our contributions

In this paper we study resettable-soundness in the standard model, rather than in the public-key model considered in [7, 30]. As was the case for resettable zero-knowledge, it is not clear a-priori whether non-trivial resettably-sound zero-knowledge protocols exist at all.

Indeed, the situation here is worse: we show that resettably-sound zero-knowledge *proofs* exist only for $\mathcal{P}/\text{poly}$.[3] Furthermore, if one is restricted to showing zero-knowledge via a black-box simulator, resettably-sound zero-knowledge *arguments* exist only for BPP-languages. Thus, our study would have come to an end, if it were not for the recent result of Barak [1] in which for the first time, zero-knowledge arguments are constructed using non-black-box simulators. This opens the door to hope to get around impossibility results regarding zero-knowledge proved via black-box simulators.

Indeed, we construct resettably-sound zero-knowledge arguments for $\mathcal{NP}$, using Barak's construction. In turn, using resettably-sound zero-knowledge arguments for $\mathcal{NP}$, we obtain two main applications:

1. Resettable zero-knowledge arguments of knowledge for $\mathcal{NP}$, using a relaxed and yet natural definition of arguments (and proofs) of knowledge (see Section 1.3.2).

2. Constant-round resettable zero-knowledge arguments for $\mathcal{NP}$, in the public-key model, under weaker assumptions than known previously: instead of assuming sub-exponential hardness, we only assume super-polynomial hardness.

[3] We also show that resettably sound proofs – without the zero-knowledge requirement – are possible only for languages in $\mathcal{NP}/\text{poly}$.

All our positive results inherit Barak's [1] intractability assumption – the existence of collision-free hash functions [2]. As the existence of collision-free hash functions implies the existence of one-way functions, we use the latter freely. Our protocols also inherit from [1] non-black-box demonstrations of various properties.

We proceed to give details on our main result and its applications.

1.3.1 Main result

Our main result is a constant-round resettably-sound zero-knowledge argument for $\mathcal{NP}$, assuming the existence of collision-free hash functions. This is achieved by showing how to transform any constant-round public-coin zero-knowledge interactive argument into a constant-round resettably-sound zero-knowledge argument for the same language, and then applying the transformation to the recent construction [1] of a constant-round public-coin zero-knowledge argument of knowledge for $\mathcal{NP}$.

Recall, that until recently, this transformation would have been useless as no constant-round public-coin zero-knowledge arguments were known for languages outside of $\mathcal{BPP}$. Indeed, Goldreich and Krawczyk proved that only languages in $\mathcal{BPP}$ have constant-round public-coin arguments and proofs that are black-box zero-knowledge [22]. Naturally, the [1] construction of a constant-round public-coin zero-knowledge argument of knowledge for $\mathcal{NP}$ must (and does) use a non-black-box simulator. Thus, we obtain:

Theorem 1.1 *If there exist collision-free hash functions, then any NP-language has a* (constant-round) *resettably-sound zero-knowledge argument. Furthermore, these protocols are arguments of knowledge.*

Using Theorem 1.1 we obtain the following applications.

1.3.2 Application 1: Resettable-ZK Arguments of Knowledge

The standard definition of an argument (or proof) of knowledge requires the knowledge-extractor to use the prover's strategy as a black-box. Furthermore, in a resetting attack on the prover, the verifier has this very same capability during the execution of the protocol . Loosely speaking, then, if the extractor can extract anything (e.g., an NP-witness) from the prover, then so can a cheating verifier mounting a resetting attack. Thus, under the standard definition (cf. [3]), resettable zero-knowledge arguments of knowledge exist only for $\mathcal{BPP}$.

We adopt a relaxation of the definition of an argument (or proof) of knowledge in which the knowledge-extractor is given the prover's program as auxiliary input (rather than given only black-box access to it). The knowledge-extractor, now, is (at least syntactically) more powerful than the cheating verifier during a resetting attack in which the latter has in essence black-box access to the prover's strategy. The relaxed definition appeared originally in Feige and Shamir [16] (which differs from the definition in [14]; see discussion in [3]), and suffices for all practical applications of arguments of knowledge.

The standard definition, allowing only oracle access to the prover's strategy, was used in the literature in the past as in principle it allows the consideration of prover strategies which are not in $\mathcal{P}$/poly (an irrelevant consideration for practical applications and for arguments in particular). Moreover, it was generally believed, that one cannot benefit from non-black-box access to the prover's code, and thus restricting access to the prover poses no limitation.

Henceforth we will use "proof of knowledge" to refer to the relaxed definition.

Using Theorem 1.1, we construct resettable zero-knowledge and resettable witness-indistinguishable arguments of knowledge for $\mathcal{NP}$. Our construction is based on a modification of a well-known design principle underlying protocols such as those in [21, 32, 7]: In these protocols, the verifier starts by committing to its queries, then the prover sends some information, and the verifier decommits to the abovementioned queries, which the prover is now supposed to answer. Such protocols usually fail to yield proofs of knowledge, as the typical way in which a knowledge-extractor works is by obtaining answers to several different queries regarding the same piece of information, but this way is blocked when the queries are committed to before the information is presented. Our modification is to replace the action of decommitment by merely revealing the committed values and proving in zero-knowledge that the revealed values are indeed those committed to. Toward this end, the verifier needs to employ a zero-knowledge proof (or argument), in which the prover plays the role of the verifier, which is why in our setting (in which the main prover is resettable) this subprotocol has to be resettably-sound. Here is where we use Theorem 1.1, which supplies us with a resettably-sound zero-knowledge argument for $\mathcal{NP}$. Thus, we obtain:

Theorem 1.2 *If there exist collision-free hash functions, then there exists*

1. *A constant-round resettable witness-indistinguishable argument of knowledge for $\mathcal{NP}$.*
2. *A poly-logarithmic round resettable zero-knowledge argument of knowledge for $\mathcal{NP}$.*

All applications of the notion of a proof of knowledge, including the Fiat-Shamir paradigm of building identification protocols from zero-knowledge (and witness indistinguishable) proofs of knowledge [17], are thus salvaged for resettable provers. This holds also with respect to constant-round protocols in the public-key model; see Theorem 1.3 (below). Of course, for any particular proof of knowledge, one needs to explicitly prove being **resettable** zero-knowledge (or witness indistinguishable).

1.3.3 Application 2: rZK in the Public-Key Model under weaker assumptions

Current protocols that achieve constant-round resettable zero-knowledge arguments for $\mathcal{NP}$, *in the public-key model*, assume a sub-exponential lower bound on the size of circuits attempting to break commitment schemes.

In contrast, using Theorem 1.1 we construct constant-round resettable zero-knowledge arguments for $\mathcal{NP}$ in the public-key model, relying only on the existence of collision-free hash functions. Thus, we replace a sub-exponential hardness assumption by a standard hardness assumption of collision-free hashing secure against all polynomial time adversaries. Furthermore, both the protocol and its analysis are conceptually simpler than the corresponding constructions presented in [7, 29].

Moreover, the constant-round protocol constructed is also an argument of knowledge (in the relaxed sense discussed in previous section).

Theorem 1.3 *If there exist collison-free hash functions, then there exists a constant-round resettable zero-knowledge argument of knowledge for $\mathcal{NP}$* in the public-key model.

We stress that previously known resettable zero-knowledge protocols (also in the public-key model) were not known to be arguments of knowledge.

1.4 Simultaneous resettability

A natural question that arises is whether it is possible to simultaneously protect both the prover and the verifier from resetting attacks. That is:

Open Problem 1.4 *Do languages outside of $\mathcal{BPP}$ have resettably-sound arguments that are resettable zero-knowledge.*

Some hope for an affirmative resolution of the above question is provided by the fact that some level of resettable-security for both parties does seem to be achievable.[4] That is:

Theorem 1.5 (implicit in [11]): *Assuming the existence of trapdoor permutations, any NP-language has a resettably-sound proof that is resettable witness-indistinguishable.*

Theorem 1.5 follows from the following facts regarding ZAPs (as defined by Dwork and Naor [11]). Loosely speaking, ZAPs are two-round public-coin witness-indistinguishable proofs. Thus, by definition, ZAPs are resettably-sound (because even in a single session the prover obtains all the verifier's coins before sending its own message). On the other hand, as noted in [11], any ZAP can be made resettable witness-indistinguishable (by using pseudorandom functions as in the transformation of [7]). Using a main result of [11], by which ZAPs for $\mathcal{NP}$ can be constructed based on any non-interactive zero-knowledge proofs for $\mathcal{NP}$, which in turn can be constructed based on trapdoor permutations (cf. [15, 5]), Theorem 1.5 follows.

We conjecture that resettably-sound resettable *zero-knowledge* arguments for $\mathcal{NP}$ do exist, and have made some progress towards establishing this.

2 Preliminaries

2.1 General Preliminaries

We briefly review some well-known notions.

Polynomial-size adversaries. We focus on polynomial-size adversaries. By this we mean adversaries that employ a strategy that can be implemented by a non-uniform family of polynomial-size circuits.

Interactive proof systems [24]. We consider computationally-sound interactive proof systems (a.k.a arguments) [4] in which it is infeasible for any polynomial-size circuit to cheat with non-negligible probability. Specifically, for every polynomial p and all sufficiently large inputs x not in the language, every circuit of size $p(|x|)$ (representing a cheating prover strategy) may convince the verifier to accept only with probability less than $1/p(|x|)$. We further restrict the meaning of the term 'interactive proof system' by requiring that inputs in the language are accepted with probability 1 (i.e., so-called *perfect completeness*).

[4] The evidence provided by Theorem 1.5 (towards an affirmative resolution of the above question) is admittingly not very strong. In general, zero-knowledge seems a significantly stronger notion of security than witness-indistinguishability. Furthermore, Theorem 1.5 yields resettably-sound proofs, whereas (as mentioned above) there is no hope of obtaining resettably-sound zero-knowledge proofs (rather than arguments) for languages outside $\mathcal{P}$/poly.

Zero-knowledge. We adopt the basic paradigm of the definition of zero-knowledge [24]: The output of every probabilistic polynomial-time adversary (verifier) that interacts with the designated prover on a common input in the language, ought to be simulatable by a probabilistic polynomial-time machine (which interacts with nobody), called the simulator. In fact, we focus on universal simulators that given the code of any polynomial-size adversary (or oracle access to its strategy) simulates (by itself and without interacting with the prover) the interaction of this adversary with the prover. In case this universal simulator only uses oracle access to the adversary's strategy (rather than being given its code) , we call it black-box.

Witness indistinguishable proof systems [16]. Loosely speaking, these are proof systems in which the prover is a probabilistic polynomial-time machine with auxiliary input (typically, an NP-witness), having the property that interactions in which the prover uses different "legitimate" auxiliary-inputs are computationally indistinguishable from each other. Recall that any zero-knowledge proof system is also witness indistinguishable, and there are witness indistinguishable proof systems that are not zero-knowledge.

2.2 Resettable Zero-Knowledge (and Witness Indistinguishability)

In this section we recall the definition of resettable zero-knowledge and witness indistinguishability. The text is adapted from [7].

Given a specified prover P, a common input x and an auxiliary input y to P (e.g., y may be an NP-witness for x being in some NP-language), we consider polynomially-many interactions with the deterministic prover strategy $P_{x,y,\omega}$ determined by uniformly selecting and fixing P's coins, denoted ω. That is, ω is uniformly selected and fixed once and for all, and the adversary may invoke and interact with many instances of $P_{x,y,\omega}$. An interaction with an instance of $P_{x,y,\omega}$ is called a session. It is stressed that $P_{x,y,\omega}$'s actions in each session are oblivious of other sessions (since $P_{x,y,\omega}$ mimics the "single session strategy" P); nonetheless, the actions of the adversary may depend on other sessions.

There are two variants of the above model. In the basic variant, a session must be terminated (either completed or aborted) before a new session can be initiated by the adversary. In the interleaving variant, this restriction is not made and so the adversary may concurrently initiate and interact with $P_{x,y,\omega}$ in many sessions. In [7], these variants were proven to be equivalent. Thus, for sake of simplicity, we focus on the simpler basic (i.e., non-interleaving) variant.

An extension to the above model is obtained by allowing the adversary to interact (many times) with several random independent incarnations of P (rather than with a single one). That is, rather than interacting many times with one $P_{x,y,\omega}$, where ω is randomly selected, the adversary many interact many times with each P_{x_i,y_i,ω_j}, where the ω_j's are independently and randomly selected. Intuitively, allowing several independent random incarnations (i.e., several ω_j's) should not increase the power of the adversary, but it is not clear whether this intuition is sound. We do know (see [7]) that allowing several different inputs (i.e., x_i's) for the same random-tape does increase the power of the adversary. Anyhow, with these extensions, resettable security implies security under concurrent executions.

Definition 2.1 (rZK and rWI - standard model): *An interactive proof system (P, V) for a language L is said to be* resettable zero-knowledge *if for every probabilistic polynomial-time adversary V^* there exists a probabilistic polynomial-time simulator M^* so that the following two distribution ensembles* *are computational indistinguishable: Let each distribution be indexed by a sequence of* distinct[5] *common inputs* $\overline{x} = x_1, ..., x_{\text{poly}(n)} \in L \cap \{0,1\}^n$ *and a corresponding sequence of prover's auxiliary-inputs* $\overline{y} = y_1, ..., y_{\text{poly}(n)}$,

`Distribution 1` *is defined by the following random process which depends on* P *and* V^*.

1. *Randomly select and fix* $t = \text{poly}(n)$ *random-tapes,* $\omega_1, ..., \omega_t$, *for* P, *resulting in deterministic strategies* $P^{(i,j)} = P_{x_i, y_i, \omega_j}$ *defined by* $P_{x_i,y_i,\omega_j}(\alpha) = P(x_i, y_i, \omega_j, \alpha)$, *for* $i, j \in \{1, ..., t\}$. *Each* $P^{(i,j)}$ *is called an* incarnation of P.
2. *Machine* V^* *is allowed to run polynomially-many sessions with the* $P^{(i,j)}$*'s. Throughout these sessions,* V^* *is required to complete its current interaction with the current copy of* $P^{(i,j)}$ *before starting a new interaction with any* $P^{(i',j')}$, *regardless if* $(i,j) = (i',j')$ *or not. Thus, the activity of* V^* *proceeds in rounds. In each round it selects one of the* $P^{(i,j)}$*'s and conducts a complete interaction with it.*
3. *Once* V^* *decides it is done interacting with the* $P^{(i,j)}$*'s, it* (i.e., V^*) *produces an output based on its view of these interactions. Let us denote this output by* $\langle P(\overline{y}), V^* \rangle(\overline{x})$.

`Distribution 2:` *The output of* $M^*(\overline{x})$.

In case there exists a universal probabilistic polynomial-time machine, M, *so that* M^* *can be implemented by letting* M *have oracle-access to* V^*, *we say that* P *is* resettable zero-knowledge via a black-box simulation.[6]

An interactive proof system (P, V) *for* L *is said to be* resettable witness indistinguishable (rWI) *if every two distribution ensembles of Type 1 that are indexed by the same sequence of distinct inputs* $\overline{x} = x_1, ..., x_{\text{poly}(n)} \in L \cap \{0,1\}^n$, *(but possibly different sequences of prover's auxiliary-inputs,* $\text{aux}^{(1)}(\overline{x}) = y_1^{(1)}, ..., y_{\text{poly}(n)}^{(1)}$ *and* $\text{aux}^{(2)}(\overline{x}) = y_1^{(2)}, ..., y_{\text{poly}(n)}^{(2)}$*), are computationally indistinguishable. That is, we require that* $\{\langle P(\text{aux}^{(1)}(\overline{x})), V^* \rangle(\overline{x})\}_{\overline{x}}$ *and* $\{\langle P(\text{aux}^{(2)}(\overline{x})), V^* \rangle(\overline{x})\}_{\overline{x}}$ *are computationally indistinguishable.*

2.3 Arguments of Knowledge with Non Black-Box Extraction

In the standard definition of proofs of knowledge (cf. [3]), the knowledge-extractor is given oracle (or black-box) access to the prover strategy. As mentioned in [7], under this definition, resettable zero-knowledge proofs (or arguments) of knowledge exist only for languages in $\mathcal{BPP}$. This is because in a resetting attack, the verifier has the same power as the extractor. Therefore, the proof (or argument) cannot be zero-knowledge (or even witness indistinguishable).

Below, we recall a natural, yet relaxed definition of arguments of knowledge where the extractor has access to the *description* of the prover strategy (cf. [16]). Thus, the extractor has (potentially) more power than even a verifier that can execute a resetting attack.

Definition 2.2 (system of arguments of knowledge, relaxed definition [16]): *Let* R *be a binary relation. We say that a probabilistic, polynomial-time interactive machine* V *is a* knowledge verifier for the relation R *with* negligible knowledge error *if the following two conditions hold:*

[5]This condition (of the inputs being distinct) was mistakenly omitted from the definition in [7], but their analysis assumes it. It seems that this requirement is essential for the non-triviality of rZK and that no protocol can achieve the stronger definition (in which some of the x_i's may be equal whereas the corresponding auxiliary y_i's may be either equal or not). In particular, all known rZK protocol (including ours) are not even rWI under the stronger definition.

[6]Recall that the existence of black-box simulators implies auxiliary-input zero-knowledge (cf. [25, 22]).

- Non-triviality: *There exists a probabilistic polynomial-time interactive machine P such that for every $(x,y) \in R$, all possible interactions of V with P (with auxiliary input y) on common input x are accepting.*
- Validity (or knowledge soundness) with negligible error: *There exists a probabilistic expected polynomial-time machine K such that for every probabilistic polynomial-time machine P^*, every polynomial $p(\cdot)$ and all sufficiently large x's*

$$\Pr[K(\mathsf{desc}(P^*), x) \in R(x)] > \Pr[\langle P^*, V\rangle(x) = \text{ACCEPT}] - \frac{1}{p(|x|)}$$

where $\langle P^*, V\rangle(x)$ denotes V's output after interacting with P^* upon common input x, $\mathsf{desc}(P^*)$ denotes the description of P^*'s strategy, and $R(x) = \{y : (x,y) \in R\}$ denotes the set of witnesses for x.

3 Resettable-Soundness

In this section we define and study various notions of resettable-soundness. Specifically, we define resettably-sound proofs and arguments, and justify our focus on the latter (where soundness holds only with respect to polynomial-size cheating provers, rather than for arbitrary cheating provers).

3.1 Definitions

We adopt the formalism of resettable zero-knowledge (cf. [7]), with the understanding that here the adversary plays the role of the prover and has the power to reset the verifier (or invoke it several times on the same sequence of coins).[7]

Given a specified verifier program V and a common input x, we consider polynomially-many interactions with the residual deterministic verifier strategy $V_{x,r}$ determined by uniformly selecting and fixing V's coins, denoted r. That is, r is uniformly selected and fixed once and for all, and the adversary may invoke and interact with $V_{x,r}$ many times. Each such interaction is called a session. Thus, the adversary and $V_{x,r}$ engage in polynomially-many sessions; but whereas $V_{x,r}$'s actions in the current session are oblivious of other sessions (since $V_{x,r}$ mimics the "single session strategy" V), the actions of the adversary may depend on other sessions. Typically, $x \notin L$ and the aim of the adversary, or cheating prover, is to convince $V_{x,r}$ to accept x in one of these sessions. (In the context of resettable zero-knowledge, the adversary is called a cheating verifier and its aim is to "extract knowledge" from the prover by possibly resetting it.)

We consider two variants of the model. In the first (and main) variant, a session must be terminated (either completed or aborted) before a new session can be initiated by the adversary. In the second (interleaving) variant, this restriction is not made and so the adversary may concurrently initiate and interact with $V_{x,r}$ in many sessions. A suitable formalism must be introduced in order to support these concurrent executions. (For simplicity, say that the adversary prepends a session-ID to each message it sends, and a distinct copy of $V_{x,r}$ handles all messages prepended by each fixed ID.) Note that in both variants, the adversary may repeat in the current session the same messages sent in a prior session, resulting in an identical prefix of an interaction (since the verifier's randomness is fixed). Furthermore, by deviating in the next message, the adversary may obtain two different continuations of the same prefix of an interaction. Viewed in other terms, the adversary

[7] In contrast, in the context of resettable zero-knowledge, the adversary plays the role of the verifier and has the power to reset the prover.

may "effectively rewind" (or "reset") the verifier to any point in a prior interaction, and carry-on a new continuation (of this interaction prefix) from this point.

For sake of simplicity, we will present only the definition of the main (non-interleaving) model. We can afford to focus on the non-interleaving model because the argument given in [7] by which the models are equivalent with respect to resettable zero-knowledge hold also with respect to resettable-soundness; the reason being that this argument merely shows how a resetting-adversary in the interleaving model can be perfectly emulated by a resetting-adversary in the non-interleaving model.

Following Canetti *et al.* [7], we extend the basic model to allow the adversary to interact (many times) with several random independent incarnations of V (rather than with a single one). That is, rather than interacting many times with one $V_{x,r}$, where r is randomly selected and x is predetermined, the adversary may interact many times with different V_{x_i,r_j}'s, where the r_j's are independently and randomly selected and the x_i's are chosen dynamically by the adversary. One may be tempted to say that the ability to interact with several incarnations of V should not add power to the model, but as shown in [7] this intuition is not valid.

One important deviation from the formalism of Canetti *et al.* [7], is in not fixing a sequence of (polynomially-many) x_i's ahead of time, but rather allowing the adversary to select such x_i's on the fly. Furthermore, adversarial selection of inputs is used in both the completeness and soundness conditions. (We comment that the latter strengthening of the definition is applicable and desirable also in the setting of resettable zero-knowledge.)

Definition 3.1 (resettable verifier – main model): *A* resetting attack *of a cheating prover P^* on a* resettable verifier *V is defined by the following two-step random process, indexed by a security parameter n.*

1. *Uniformly select and fix $t = \mathrm{poly}(n)$ random-tapes, denoted $r_1, \ldots, r_t$, for V, resulting in deterministic strategies $V^{(j)}(x) = V_{x,r_j}$ defined by $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$, where $x \in \{0,1\}^n$ and $j \in [t]$.*[8] *Each $V^{(j)}(x)$ is called an* incarnation of V.
2. *On input 1^n, machine P^* is allowed to initiate $\mathrm{poly}(n)$-many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0,1\}^n$ and $j \in [t]$, thus defining $V^{(j)}(x)$, and conducts a complete session with it.*

Let P and V be some pair of interactive machines, and suppose that V is implementable in probabilistic polynomial-time. We say that (P, V) is a resettably-sound proof system for L (resp., resettably-sound argument system for L) *if the following two conditions hold:*

- Resettable-completeness: *Consider an arbitrary resetting attack* (resp., polynomial-size resetting attack)*, and suppose that in some session, after selecting an incarnation $V^{(j)}(x)$, the attacker follows the strategy P.*[9] *Then, if $x \in L$ then $V^{(j)}(x)$ rejects with negligible probability.*
- Resettable-soundness: *For every resetting attack* (resp., polynomial-size resetting attack)*, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L$ is negligible.*

[8] Recall that $V(x, r, \alpha)$ denotes the message sent by the strategy V on common input x, random-tape r, after seeing the message-sequence α.

[9] In fact, in order to consider honest prover strategies that are implementable in probabilistic polynomial-time, we need to supply P with an adequate NP-witness. That is, let R be an NP-relation that corresponds to the NP-language L. Then we consider a resetting attack that for every selected $x \in L$ also provides P with (an NP-witness) w satisfying $(x, w) \in R$. In this case, we require that when $V^{(j)}(x)$ interacts with $P(w)$ it rejects with negligible probability.

We stress that by a resettably-sound *proof* we mean that the resettable-soundness requirement holds also for computationally unbounded cheating provers, whereas only polynomial-size cheating provers are considered in the definition of resettably-sound *arguments*.

We also adapt the definition of a proof of knowledge to the resettable context. We assume that the reader is familiar with the basic definition of a proof of knowledge (cf., [3]). The basic approach is to link the probability that any prover strategy convinces the verifier to the efficiency of extracting the claimed knowledge by using this prover strategy as an oracle. Thus, a definition of a resettably-sound proof (or argument) of knowledge should refer to the probability of convincing the verifier during a resetting attack (rather than to the probability of convincing the verifier in an ordinary interaction). On the other hand, we relax the definition by allowing the extractor to depend on the size of the prover strategy (i.e., we focus on polynomial-size provers and allow a different extractor per each polynomial size-bound).[10] The latter relaxation seems important for our positive results and is certainly sufficient for our applications (as well as for any other application we can think of). For simplicity, we consider below only NP-relations.

Definition 3.2 (resettably-sound argument of knowledge, sketch): *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP-relation for an NP-language $L = \{x : \exists w\, (x,w) \in R\}$. We say that (P,V) is a* resettably-sound argument of knowledge for R *if*

- *(P,V) is a resettably-sound argument for L, and*
- *for every polynomial q there exists a probabilistic expected polynomial-time oracle machine E such that for every resetting attack P^* of size $q(n)$, the probability that $E^{P^*}(1^n)$ outputs a witness for the input selected in the last session is at most negligibly smaller than the probability that P^* convinces V in the last session.*

The focus on the last session of the resetting attack is done for simplicity and is valid without loss of generality (because the prover can always duplicate the interaction of any session of its choice in the last session).

Important Note: We stress that, in the rest of this section, zero-knowledge mean the standard notion (rather than *resettable zero-knowledge*).

3.2 Limitations of resettably-sound proofs

In this subsection we justify our focus on resettably-sound *arguments* (rather than resettably-sound *proofs*): As demonstrated below, resettably-sound proofs exist only in an almost trivial manner, and more annoyingly resettably-sound *zero-knowledge* proofs exist only for languages having (non-uniform) polynomial-size circuits (and thus are unlikely to exist for all $\mathcal{NP}$).

Theorem 3.3 *Suppose that there exists a resettably-sound proof for L. Then, L is contained in non-uniform $\mathcal{NP}$* (i.e., $L \in \mathcal{NP}/\mathrm{poly}$). *Furthermore, if this proof system is zero-knowledge then L is contained in non-uniform polynomial-time* (i.e., $L \in \mathcal{P}/\mathrm{poly}$).

[10] In fact, it suffices to allow a different extractor per each polynomial bound on the number of sessions initiated by the prover. (Such a relaxation coincides with the standard definition for the case of a single session.) However, since we focus on polynomial-size provers, we may as well refer to their size.

Note that $\mathcal{AM} \subset \mathcal{NP}/\text{poly}$ does have resettably-sound proof systems (e.g., the first message sent in a properly amplified AM-proof system can be used to correctly prove membership of all strings of adequate length).[11] Similarly, $\mathcal{BPP} \subset \mathcal{P}/\text{poly}$ does have resettably-sound zero-knowledge proof systems (in which the verifier just decides by inspecting the input). We believe that Theorem 3.3 holds also with $\mathcal{NP}/\text{poly}$ replaced by $\mathcal{AM}$ and $\mathcal{P}/\text{poly}$ replaced by $\mathcal{BPP}$.

Proof Sketch: Intuitively, the verifier's randomness is a-priori bounded, whereas the number of sessions in which it takes place (in a resettable attack) is not a priori bounded. Thus, there must be a session in which the verifier uses very little truly new randomness (i.e., there is a session in which the verifier's moves are almost determined by the history of previous sessions). Loosely speaking, the limitations of deterministic verifiers (with respect to interactive proofs and zero-knowledge proofs; cf., [18] and [25], respectively) should apply here. The actual proof is more complex; see below.

We start with the main part of the theorem: Suppose that (P, V) is an interactive proof as in the main part of the theorem, and suppose that on common input x machine V uses a random-tape of length $m = \text{poly}(|x|)$. We consider a (generic) deterministic cheating (resetting) prover that interacts with V for several sessions. Each possible execution sequence of i sessions determines a set of possible verifier tapes that are consistent with the verifier's actions in these i sessions. We say that a certain transcript of i sessions is critical if there is no way for the prover to interact in the $i+1^{\text{st}}$ session so that the possible sets determined by the $i+1$ session-executions are all smaller than 99% of the size of the set determined by the i first sessions. (The above prover's actions include the selection of a common input of adequate length.)

Let us consider such a critical transcript, and denote by S the set of possible verifier tapes that are consistent with the verifier's actions in this transcript. In fact, we consider a critical transcript and a corresponding set S so that completeness and soundness of the next session hold also when the verifier's random-tape is uniformly distributed in S.[12] The actions of V during the next session can be (almost) determined ahead of time by the cheating prover (which, being computationally unbounded, may uniformly select $s \in S$ and act according to the tape s). Thus, given the critical transcript, one may provide "almost an NP-proof" for membership in L. It is instructive to think of S as a singleton, in which case the prover can supply NP-proofs for membership in L. In general, when S is not a singleton, we construct a non-uniform constant-round proof system as follows. The new (non-uniform) verifier incorporates in its code a critical transcript α (possibly of several sessions) and the cardinality of the set S of the corresponding tapes of V that are consistent with this transcript. On common input x, the new prover uniformly selects $s \in S$ and sends to the verifier the corresponding transcript of the next session that is consistent with the actions of V on input x and random-tape s. Next, the parties execute a (constant-round) random selection protocol so that the new verifier obtains an almost-random $r \in S$. (The new verifier uses the critical transcript α and $|S|$ in order to execute its part in the protocol, which requires the verifier to know $|S|$ and to be able to decide membership in S. The fact that the prover is computationally unbounded is necessary for playing its role in the random selection protocol.) Finally, the new verifier accepts x if and only if the the transcript of the current session (sent by the prover) is consistent with the actions of V on input x and random-tape r (rather than s). Observe that the new interactive proof emulates almost perfectly the execution of the next session of the original proof system (under a resetting attack). Using the hypothesis that completeness and soundness of the next session (played by V) hold also when the verifier's random-tape is uniformly distributed in S, it follows that the

[11] Recall that $\mathcal{AM}$ stands for the class of languages having two-round public-coin interactive proofs.

[12] The last sentence makes sense only when considering computationally-unbouded cheating provers.

new proof system satisfies the completeness and soundness properties. Thus, L has a non-uniform constant-round proof system. Using known results about constant-round interactive proofs (i.e., $\mathcal{IP}(O(1)) \subseteq \mathcal{AM}$ and $\mathcal{AM} \subset \mathcal{NP}/\text{poly}$), the first part of the theorem follows.

We now turn to the second part of the theorem, and consider the case where the above (resettably-sound) proof system is zero-knowledge. We consider the critical transcript α and set S as defined above. Recall that the completeness and soundness of the next session hold also when the verifier's random-tape is uniformly distributed in S. Fixing two random strings $s, r \in S$, and using the simulator guaranteed by the (ordinary) zero-knowledge condition of (P, V), we obtain a non-uniform polynomial-size circuit that decides whether or not x (of suitable length) is in L as follows. The circuit, which incorporates the strings s and r, uses the original simulator to generate a transcript of an interaction of V' with P, where V' behaves as V on input x and coins s. The circuit accepts if and only if the generated transcript is accepting and consistent with the actions of V on input x and coins r (rather than coins s). Using the hypothesis that completeness and soundness of the next session (played by V) hold also when the verifier's random-tape is uniformly distributed in S, it follows that the circuit decides correctly. ■

3.3 On the triviality of resettably-sound black-box zero-knowledge

In this section we explain why the resettably-sound zero-knowledge arguments presented in the next subsection are not accompanied (as usual) by a black-box simulator. Specifically, we show that only a language in $\mathcal{BPP}$ can have a resettably-sound argument with a black-box zero-knowledge simulator (and, in fact, $\mathcal{BPP}$ languages have trivial "proof" systems in which the prover does not even take part). Independently, Reyzin in [31] showed that Theorem 3.4 holds even in the public-key model.

Theorem 3.4 *Suppose that there exists a resettably-sound argument for L, and that this protocol is black-box zero-knowledge. Then, $L \in \mathcal{BPP}$.*

Proof Sketch: Intuitively, a (probabilistic polynomial-time) cheating prover mounting a resettable attack on the verifier, may emulate the actions of the black-box simulator (with access to the deterministic verifier defined by fixing the random-tape). Thus, in case $x \in L$, this cheating prover causes the verifier to accept x (because the emulated simulator succeeds in producing an accepting conversation). On the other hand, by the resettable-soundness condition, the cheating prover is unlikely to cause the verifier to accept $x \notin L$. Finally, observing that the cheating prover described above is implementable in probabilistic polynomial-time, we obtain a probabilistic polynomial-time decision procedure for L. Details follows.

Let (P, V) be a resettably-sound argument for L, and let M be a (probabilistic polynomial-time) black-box simulator such that for every family of polynomial-size circuits $\{V_x\}_{x \in L}$ the distributions $\{\langle P, V_x \rangle(x)\}_{x \in L}$ and $\{M^{V_x}(x)\}_{x \in L}$ are computationally indistinguishable, where $\langle P, V_x \rangle(x)$ means the transcript of an ordinary interaction between (non-resettable) interactive machines. Using M, we construct a cheating prover P^* that operates in the model of resettable-soundness and thus may reset the verifier. (In fact, P^* uses its input, x, in all sessions it conducts with V_x, rather than selecting adaptively an input for each session.) The cheating prover $P^*(x)$ just emulates the actions of $M(x)$, while treating the (resettable) verifier as an oracle. (Recall that each query to the oracle corresponds to a sequence of prover messages, and so obtaining the corresponding answer amounts to initiating a new session with V_x and sending messages as in the given sequence/query.) We will focus on the last session in the interaction of the cheating prover with the verifier, which corresponds to the output produced by M (since, w.l.o.g., M performs a full session corresponding to its output before writing-out its output).

For every common-input x and random-input r (for V), we consider the (deterministic) verifier strategy $V_{x,r}$ (as in Definition 3.1). By the completeness condition, for every $x \in L$ and almost all r's, with high probability, the strategy $V_{x,r}$ accepts x when interacting with (the honest prover) P. Thus, with high probability over the choices of r and the internal coin tosses of M, it is the case that $M^{V_{x,r}}(x)$ is an accepting transcript (of a single session). It follows that (with high probability over the internal coin tosses of V and of the cheating resetting prover P^*), the last session in the interaction of $V_{x,r}$ with P^* on common input $x \in L$ is convincing. On the other hand, by the resettable-soundness condition, with high probability, every session of V (with any feasible resetting cheating prover) on common input $x \notin L$ is not convincing. Thus, with very high probability, the last session in the interaction of $V_{x,r}$ with P^* on common input $x \notin L$ is not convincing.

Combining the (probabilistic polynomial-time) cheating prover P^* with V, we obtain a probabilistic polynomial-time decision procedure for L: On input x, we select a random r and emulate the interaction of the cheating prover P^* with $V_{x,r}$. We accept x if and only if the last session in this interaction is accepting. ■

3.4 How to construct resettably-sound zero-knowledge arguments

The main result of this section is obtained by combining the following transformation with a recent result of Barak [1].

Proposition 3.5 (a transformation): *Let $L \in \mathcal{NP}$ and R be a corresponding witness relation. Suppose that (P,V) is a* constant-round public-coin *argument of knowledge for R, and let $\{f_s : \{0,1\}^* \to \{0,1\}^{|s|}\}$ be a collection of pseudorandom functions. Assume, without loss of generality, that on common input x, in each round, the verifier V sends a uniformly distributed $|x|$-bit string. Let W_s be a deterministic verifier program that, on common input $x \in \{0,1\}^{|s|}$, emulates V except that it determines the the current round message by applying f_s to the transcript so far. Let W be defined so that on common input x and uniformly random-tape $s \in \{0,1\}^{|x|}$, it acts as $W_s(x)$. Then:*

1. *(P,W) is a resettably-sound argument for L. Futhermore, (P,W) is a resettably-sound argument of knowledge for R.*
2. *If (P,V) is zero-knowledge then so is (P,W). Furthermore, if the simulator of (P,V) runs in strict polynomial-time, then so does the simulator of (P,W).*
3. *If (P,V) is witness-indistinguishable then so is (P,W).*

Recall that only languages in $\mathcal{BPP}$ have a constant-round public-coin zero-knowledge argument with a black-box simulator. Thus, Proposition 3.5 may yields something interesting only when applied to protocols that do not have a black-box simulator.[13] Recall that no such (non-trivial) zero-knowledge arguments were known until very recently. Here is where the result of Barak [1]

[13] In particular, we may apply Proposition 3.5 to some known constant-round public-coin interactive proofs that are known to be witness-indistinguishable (e.g., parallel repetitions of the basic zero-knowledge proof of Goldreich, Micali and Wigderson [23]). This yields witness-indistinguishable arguments that are resettably-sound. However, for witness-indistinguishable protocols, stronger results are known; see Section 1.4. Thus, we focus below on zero-knowledge protocols.

plays a role: Using his recent constant-round public-coin zero-knowledge argument of knowledge for $\mathcal{NP}$ (which indeed uses a non-black-box simulator), we obtain resettably-sound zero-knowledge arguments (of knowledge) for $\mathcal{NP}$. This establishes Theorem 1.1.

Proof Sketch for Proposition 3.5: Parts 2 and 3 follows immediately because the zero-knowledge condition (as well as the witness-indistinguishability condition) does not refer to the honest verifier (which is the only thing we have modified) but rather to all possible polynomial-size adversaries (representing cheating verifiers).

As a preliminary step towards proving Part 1, we consider an imaginary verifier (denoted W_F) that, on common input x, uses a truly random function $F : \{0,1\}^* \to \{0,1\}^{|x|}$ (rather than a pseudorandom function f_s, for $|s| = |x|$). Loosely speaking, by the definition of pseudorandomness, all non-uniform polynomial-size provers must behave in essentially the same way under this replacement.

(The actual proof is slightly more subtle than one may realize because in our context this "behavior" is the ability to convince the verifier of the membership in L of $x \notin L$ that is chosen on the fly by the prover. The problem is how will the distinguisher determine that this event took place (notice that the distinguisher itself may not necessarily know whether or not $x \in L$). Thus, we actually proceed as follows. First, we show (below) that P' may convince W_F to accept an input not in L only with negligible probability. Next, we use the hypothesis that (P, V) is an argument of knowledge of an NP-witness, to extract such witnesses for every input accepted by W_F. Specifically, for any input accepted by W_F we may employ the knowledge-extractor to to a related (non-resetting) P'' (defined below) and obtain an NP-witness. Thus, for the (P', W_F) transcript, we expect to couple each accepted input with a corresponding NP-witness. If this does not occur with respect to the (P', W_s) transcript (where a pseudorandom function), then we distinguish a random function from a pseudorandom one. Otherwise, we are done (because when a random function is used all accepted inputs will be coupled with NP-witness guaranteeing that they are indeed in L.)[14]

We claim that for any polynomial-size cheating prover P' that convinces the resettable-verifier W_F to accept some common input x with probability ϵ, there exists a polynomial-size cheating prover P'' that convinces the original (non-resettable) verifier V to accept the same x with probability at least ϵ/m^c, where m is a bound on the number of messages sent by the prover P' and c is the number of rounds in the original protocol. Furthermore, we shall show how to transform a cheating prover for the resettable-verifier setting into a cheating prover for the standard setting of interactive proofs. This will establish Part 1 (both its main claim and the furthermore-clause).

For sake of simplicity, we shall assume that the original cheating prover P' tries to convince the resettable-verifier W_F to accept a fixed x, and only invokes a single incarnation of W_F (and does so on common input x). The argument extends easily to the general case in which P' invokes multiple incarnations of W_F and selects the common inputs adaptively.

The new cheating prover, denoted P'', tries to convince the original (non-resettable) verifier V to accept x, while emulating the actions of a cheating prover P' that may reset the imaginary resettable verifier W_F. The new cheating prover P'' proceeds as follows: It uniformly selects $i_1, ..., i_c \in \{1, ..., m\}$, and invokes (the resetting prover) P' while emulating an imaginary verifier W_F as follows. If the prefix of the current session transcript is identical to a corresponding prefix of

[14]The above text suffices for establishing the main part of Part 1. To establish the furthermore-part, we employ analogous reasoning with respect to the event that P' convinces W_F (or W_s) to accept an input without "knowing" a corresponding NP-witness, where "knowing" means that our extraction succeeds. As before, this event occurs with negligible probability when P' interacts with W_F and therefore the same must hold with respect to the interaction of P' with W_s.

a previous session, then P'' answers by copying the same answer it has given in the previous session (to that very same session transcript prefix). If (in the current session) P' sends along a message that together with the previous messages of the current session forms a new transcript prefix (i.e., the prefix of the current session transcript is different from the prefixes of all prior sessions), then P'' answers according to the following two cases:

1. The index of the current message of P' does not equal any of the c integers $i_1, ..., i_c$ selected above. In this case, P'' provides P' with a uniformly selected $|x|$-bit long string.

2. Otherwise (i.e., the index of the current message of P' equals one of the c integers $i_1, ..., i_c$), P'' forwards the current message (of P') to V and feeds P' with the message it obtains from V. (We stress that these are the only c messages of P' for which the emulation involves interaction with V.)

In both cases, the message passed to P' is recorded for possible future use.

Clearly, for any possible choice of the integers $i_1, ..., i_c$, the distribution of messages seen by P' when P'' emulates an imaginary verifier is identical to the distribution that P' sees when actually interacting with such an imaginary verifier. The reason being that in both cases different prefixes of session transcripts are answered with uniformly and independently distributed strings, while session transcripts with identical prefixes are answered with the same string. (Observe that indeed this emulation is possible because the original verifier is of the public-coin type, and thus it is possible to efficiently emulate the next verifier message.)[15]

Towards the analysis, we call a message sent by P' novel if together with the previous messages of the current session it forms a new transcript prefix (i.e., the prefix of the current session transcript is different from the prefixes of all prior sessions). Recall that the novel messages are exactly those that cause P'' to pass along (to P') a new answer (rather than copying an answer given in some previous session). The UrMessage[16] of a non-novel message is the corresponding message that appears in the first session having a transcript-prefix that is identical to the current session transcript-prefix. That is, the answer to the UrMessage of a (non-novel) message is the one being retrieved from memory in order to answer the current message. The UrMessage of a novel message is just the message itself. Using this terminology, note that the new prover P'' succeeds in cheating V if the chosen integers $i_1, ..., i_c$ equal the indices (within the sequence of all messages sent by P') of the c UrMessages that correspond to the c messages sent in a session in which P' convinced the imaginary verifier. Since with probability ϵ such a convincing session exists, P'' succeeds provided it has guessed its message indices (i.e., c indices out of m). ■

4 Constructing rWI and rZK Protocols: A paradigm revisited

A general paradigm for constructing rWI (resettable witness indistinguishable) and rZK (resettable zero-knowledge) protocols was presented in [7]. They considered a certain class of proof systems, called admissible proof systems, and defined a slight strengthening of the concurrent model, called the hybrid model. Next, they presented a transformation applicable to admissible proof systems and showed that if the original proof system is admissible and WI (respectively, ZK) in the hybrid model, then the transformed proof system is rWI (respectively, rZK).

[15] In contrast, if the verifier were not of the public-coin type, then generating a next verifier message might have required to invert an arbitrary polynomial-time computable function (mapping the verifier's coins to its first message.)

[16] We use the German prefix *Ur*, which typically means *the most ancient version of*.

In this section, we generalize the class of admissible proof systems and show that for this more general class, the transformation of [7] also holds. We use this generalization for showing that the admissible hybrid WI argument of knowledge presented in Section 5 yields a rWI argument of knowledge. (We note that this generalization is necessary, since the argument of knowledge of Section 5.1 is *not* admissible by the definition of [7].)

4.1 The Generalized Class of Admissible Protocols

Intuitively, as in [7], we consider protocols (P, V) in which the first verifier-message "essentially determines" all its subsequent messages. That is, the only freedom retained by the verifier after sending its first message is either to abort (or act so that the prover aborts) or to send a practically predetermined message.

We formalize the above intuition as follows. After the first verifier-message, called the determining message, each subsequent message of the verifier is categorized as either a *main message* or as a message belonging to an *authenticator module*. Every main message is followed by an authenticator module and the role of this module is to assure the prover that the main message sent is consistent with the determining message. In the special case that the first verifier-message is a commitment and the main verifier-message is the revealed value, a possible authenticator module is simply the (single) message consisting of the extra decommitment information that establishes the validity of this revealed value. However, in general, the authenticator module may be a protocol consisting of a number of messages (note that unlike the main message, these messages may not be determined by the verifier's determining message). For example, rather than sending the decommitment information, the verifier may authenticate the revealed value by proving (say, in zero-knowledge) that this value is indeed the value committed to in the determining message. In this case, the authenticator module contains more than one message and varies depending on the randomness of both the prover and verifier.

In an admissible protocol (as defined below), the verification of the authenticator module depends only on the prover's random-tape, the first verifier-message and the accompanying main message. (That is, the prover does not consider any other messages sent during the protocol when checking the validity of an authenticator module.) Furthermore, the prover's subsequent actions in the rest of the protocol must depend solely on whether the authenticator module is accepted (otherwise it aborts), and in case it is accepted the subsequent actions must depend only on the main message.

We first set some useful conventions regarding the presentation of protocols in the concurrent and resettable settings. The first message in a session is always sent by the verifier and specifies an incarnation of P. The second message is sent by the prover, and is called the prover initialization message. (In our protocol below, this message will actually be empty; however we include it in order to cover the class of admissible protocols defined in [7].[17]) As mentioned above, the third message, sent by the verifier, is called the determining message of the session. (By our convention, the determining message includes the previous two messages.) This terminology will become self-explanatory below.

Definition 4.1 (admissible proof-systems): *A proof-system (P, V) is called* admissible *if the following requirements hold:*

1. *The prover P consists of two parts, P_1, P_2. Similarly, the prover's random input ω is partitioned into two disjoint parts, $\omega^{(1)}, \omega^{(2)}$, where $\omega^{(i)}$ is given to P_i. The prover initialization*

[17] This prover initialization message is important for the constructions shown in [7].

message is sent by P_1.

2. *Each verifier message (other than the first one) is first received by* P_1. *If the message is a* main message, *then* P_1 *interprets the following messages as belonging to an* authenticator module. P_1 *decides whether to accept the authenticator module or to abort based on the first verifier-message (called the* determining message*), the main message and the transcript of the authenticator module itself.*[18] *If* P_1 *accepts, it forwards the* main message *to* P_2, *who generates the next prover message.*

3. *Let* V^* *be an arbitrary* (deterministic) *polynomial-size circuit, where* V^* *may execute a resetting-attack on* P *(as described in Distribution 1 of Definition 2.1). Recall that all messages sent by* V^* *to* P *are prepended by the full transcript of messages sent so far. Thus, in particular, the prefix of any message sent by* V^* *contains the determining message. Now, let* m *be some determining message and let* V^* *interact with some incarnation of* $P = (P_1, P_2)$. *Then, except with negligible probability,* V^* *is unable to generate two different main messages for some round* ℓ *whose prefix contains the same determining message* m, *and yet* P_1 *accepts both.*

There are several differences between our generalized definition of admissible protocols and the definition of [7]. Firstly, in [7], the authenticator is limited to being a single message, rather than a module. Furthermore, P_1's decision whether or not to accept the authenticator is not dependent on its randomness $\omega^{(1)}$ (and so is universally verifiable). Finally, in [7], the requirement of item (3) is stated with respect to V^* who does not have the capability of executing a resetting-attack on P.[19] However, here this (stronger) requirement is needed for the transformation.

4.2 The hybrid model and the CGGM transformation

The hybrid model. We consider the same hybrid model as that defined in [7] and recall the definition here. Loosely speaking, the hybrid model is a model of attacks that stands somewhere between the concurrent and resettable models. That is, in this model the verifier is given the ability to "partially" reset the prover (while otherwise interacting in a concurrent setting). More specifically, the difference between the concurrent model and the resettable model is that in the resettable model the "cheating verifier" V^* can invoke many incarnations of the prover with the same random input ω, whereas in the concurrent model any two incarnations of the prover have independently chosen random inputs. The hybrid model is defined for admissible protocols as defined above (where the random input of the prover is of the form $\omega = \omega^{(1)}, \omega^{(2)}$) and provides the following intermediate power to V^*. Here V^* can invoke many incarnations of the prover with the same value of $\omega^{(1)}$; but any two incarnations of the prover must have independently chosen values for $\omega^{(2)}$. Thus, the randomness used in the prover initialization message and the verification of the authenticator modules may be reused (as in the resettable setting). On the other hand, the randomness used for the other prover messages (i.e., those determined by P_2) is fresh for each session (as in the concurrent setting).

[18] We stress that P_1's decision may depend on the randomness $\omega^{(1)}$. Thus, unlike in [7], we do not require that the validity of the verifier's messages be universally verifiable (but rather may be verifiable only by the prover).

[19] In the context of [7], if the prover specified by the protocol definition is probabilistic polynomial-time, then there is no difference between the power of an adversarial V^* who can reset P and one who cannot. That is, the transformation from admissible hybrid ZK (resp., WI) to rZK (resp., rWI) holds either way. This is due to the universal verifiability of the authenticators defined there. However, in our case, where the authentication may be interactive and dependent upon the prover's randomness, it is crucial to the transformation that in an admissible protocol, V^*'s main messages are determined even if it is able to reset the prover.

More formally, in admissible proof systems an incarnation of the prover is identified via three indices: $P^{(i,j,k)} = P_{x_i,y_i,\omega_{j,k}}$, where $\omega_{j,k} = \omega_j^{(1)}, \omega_k^{(2)}$. That is, i specifies the input, j specifies the random input to P_1 and k specifies the random input to P_2.

Definition 4.2 (hZK and hWI): *A* hybrid cheating verifier V^* *works against admissible proof systems as described above. That is, V^* proceeds as in Distribution 1 of Definition 2.1 with the exception that V^* cannot interact with incarnations $P^{(i,j,k)}$ and $P^{(i',j',k')}$ such that $k = k'$. An admissible proof system is* hZK *(resp.,* hWI*) if it satisfies Definition 2.1 with respect to hybrid cheating verifiers.*

The transformation. We now recall the transformation of [7] from admissible proof systems to resettable ones:

Construction 4.3 *Given an admissible proof system (P, V), where $P = (P_1, P_2)$, and a collection $\{f\}$ of pseudorandom functions (see [20]), we define a new proof system $(\mathbf{P}, \mathbf{V})$ as follows.*

The new verifier *is identical to V.*

The new prover: *The new prover's randomness is viewed as a pair $(\omega^{(1)}, f)$, where $\omega^{(1)} \in \{0,1\}^{\mathrm{poly}(n)}$ is of length adequate for the random-tape of P_1, and $f : \{0,1\}^{\leq \mathrm{poly}(n)} \to \{0,1\}^{\mathrm{poly}(n)}$ is a description of a function taken from an ensemble of pseudorandom functions. For convenience we describe the new prover, $\mathbf{P}$, as a pair $\mathbf{P} = \mathbf{P}_1, \mathbf{P}_2$. $\mathbf{P}_1$ is identical to P_1 with random-tape $\omega^{(1)}$; $\mathbf{P}_2$ emulates the actions of P_2 with a random tape that is determined by applying f to the determining message and the input. That is, upon receiving the determining message, denoted* `msg`*, $\mathbf{P}_2$ sets $\omega^{(2)} = f(x, \omega^{(1)}, \mathtt{msg})$ and runs P_2 with random input $\omega^{(2)}$. From this step on, $\mathbf{P}_2$ emulates the actions of P_2 using $\omega^{(2)}$ as P_2's random-tape.*

4.3 Validity of the transformation w.r.t the generalized class

We now prove that the above transformation suffices for achieving resettable witness indistinguishability and resettable zero-knowledge. Our proof is similar in spirit to that of [7], with some crucial differences. In particular, our treatment of main messages and their corresponding authenticator modules is completely different (and significantly more complex).

Theorem 4.4 *Suppose that (P, V) is admissible, and let $\mathbf{P}$ be the prover strategy obtained from P by applying Construction 4.3. Then:*

- *Assuming that pseudorandom functions exist, for every probabilistic polynomial-time resetting cheating verifier $\mathbf{V}^*$ (as in Definition 2.1) there exists a probabilistic polynomial-time hybrid cheating verifier W^* (as in Definition 4.2) so that $\langle P(\overline{y}), W^* \rangle(\overline{x})$ is computationally indistinguishable from $\langle \mathbf{P}(\overline{y}), \mathbf{V}^* \rangle(\overline{x})$.*
- *If (P, V) is a proof (or argument) of knowledge, then so too is $(\mathbf{P}, \mathbf{V})$.*

Corollary 4.5 *For (P, V) and $(\mathbf{P}, \mathbf{V})$ as in Construction 4.3, if (P, V) is* hWI *then $(\mathbf{P}, \mathbf{V})$ is* rWI*. Similarly, if (P, V) is* hZK *then $(\mathbf{P}, \mathbf{V})$ is* rZK.

Proof of Theorem 4.4 (sketch): Our analysis refers to a mental experiment in which $\mathbf{P}$ utilizes a truly random function rather than a pseudorandom one. As usual, the corresponding views of the verifier $\mathbf{V}^*$ in the two cases (i.e., random versus pseudorandom function) are computationally indistinguishable. From this point on, we identify the random-tape of $\mathbf{P}$ with a truly random function. Recall that $\langle \mathbf{P}(\overline{y}), \mathbf{V}^* \rangle(\overline{x})$ denotes the view (or output) of $\mathbf{V}^*$ after interacting with $\mathbf{P}$ on various inputs under the *resettable* model. Similarly, $\langle P(\overline{y}), W^* \rangle(\overline{x})$ denotes the view (or output) of W^* after interacting with P on various inputs under the *hybrid* model.

We construct a hybrid-model adversary, W^*, that interacts with incarnations of P, denoted $P^{(i,j,k)}$ (as in Def. 4.2). To satisfy Definition 4.2, this W^* will invoke each $P^{(i,j,k)}$ at most once, and furthermore if it invokes $P^{(i,j,k)}$ then it will not invoke any other $P^{(i',j',k)}$. Essentially, W^* serves as a "mediator" between adversary $\mathbf{V}^*$ and (the incarnations) of the prover P. That is, W^* runs $\mathbf{V}^*$; whenever $\mathbf{V}^*$ starts a new session whose determining message is different from all previous ones, W^* merely relays the messages of this session between $\mathbf{V}^*$ and P. When $\mathbf{V}^*$ "replays" an existing session s (i.e., $\mathbf{V}^*$ starts a new session whose determining message is identical to that of an existing session s), W^* responds to $\mathbf{V}^*$ using the answers of P in session s, *without interacting with* P. Finally W^* outputs whatever $\mathbf{V}^*$ outputs.

The construction of W^*. Working in the hybrid model, W^* handles the messages of $\mathbf{V}^*$ as follows (we note that the handling of authenticator messages sent by $\mathbf{V}^*$ is described in Item (4) and this is the only deviation from the description in [7]):

1. $\mathbf{V}^*$ *initiates a new session with some* $\mathbf{P}^{(i,j)}$: In this case W^* initiates a new session with $P^{(i,j,k)}$, where k is a new index not used so far. Next it obtains the prover initialization message, and forwards it to $\mathbf{V}^*$.

   We stress that a session with $P^{(i,j,k)}$ may be invoked (in the hybrid model) even if a session with some $P^{(i,j,k')}$, with $k' \neq k$, was invoked before. In the latter case, since the randomness $\omega_j^{(1)}$ is identical in both sessions, the prover initialization message obtained from $P^{(i,j,k)}$ is identical to the prover initialization message obtained previously from $P^{(i,j,k')}$.

2. $\mathbf{V}^*$ *sends a new determining message to* $\mathbf{P}^{(i,j)}$: That is, we refer to the case where $\mathbf{V}^*$ sends a determining message in the current session, and assume that this message is different from all determining messages sent in prior sessions with $\mathbf{P}^{(i,j)}$. Let $\texttt{msg}$ denote the message sent by $\mathbf{V}^*$. Then W^* sends $\texttt{msg}$ to one of the sessions of the form $P^{(i,j,\cdot)}$ that still awaits a determining message, obtains the response, and forwards it to $\mathbf{V}^*$. It designates this session (with $\mathbf{P}^{(i,j)}$) as the active session of $(i, j, \texttt{msg})$, and stores the prover's response.

   (All subsequent sessions of $\mathbf{V}^*$ with $\mathbf{P}^{(i,j)}$ in which the determining message equals $\texttt{msg}$ will be "served" by the single session of W^* designated as the active session of $(i, j, \texttt{msg})$.)

3. $\mathbf{V}^*$ *repeats a first-message to* $\mathbf{P}^{(i,j)}$: That is, we refer to the case where the current message sent by $\mathbf{V}^*$ is the determining message in the current session, and assume that this message equals a determining message, $\texttt{msg}$, sent in a prior session of $\mathbf{V}^*$ with $\mathbf{P}^{(i,j)}$. In this case, W^* retrieves from its storage P's answer in the active session of $(i, j, \texttt{msg})$, and forwards it to $\mathbf{V}^*$.

   We stress that W^* does not communicate with any session of P in this case. (Note that if W^* were to send the same message $\texttt{msg}$ to two sessions of the form $P^{(i,j,\cdot)}$ then the responses could have differed, whereas $\mathbf{V}^*$ expects to see exactly the same answer in sessions with $\mathbf{P}^{(i,j)}$ in which it sends the same $\texttt{msg}$.)

4. $\mathbf{V}^*$ *sends a main message to* $\mathbf{P}^{(i,j)}$: That is, we refer to the case where $\mathbf{V}^*$ sends a main message in the current session with $\mathbf{P}^{(i,j)}$. The key point here is to ensure that W^* forwards the message to P in the active session of $(i, j, \texttt{msg})$ only if the message is valid. Actually,

the requirement is even more strict and demands that W^* forwards the message to P only if P itself would accept the authenticator. We stress that W^* must NOT forward to an active session of P, any invalid main message of $\mathbf{V}^*$ (or, likewise messages from an authenticator module that P would reject). Otherwise, P would close the active session and W^* would not be able use it in order to handle a corresponding valid message or accepting authenticator module that may be sent by $\mathbf{V}^*$ in a future session.[20]

Therefore, at this point W^*'s aim is to check whether or not P, *in the active session of* $(i, j, \texttt{msg})$, will accept the main message and authenticator module from $\mathbf{V}^*$. Machine W^* does this as follows: W^* invokes a new session with $P^{(i,j,k)}$, where k is a new index not used so far. We stress that this invocation is used only by W^* and messages from it are never passed to $\mathbf{V}^*$. Then, after invoking $P^{(i,j,k)}$, machine W^* sends it the verifier's determining message $\texttt{msg}$. Next, W^* replays every message to $P^{(i,j,k)}$, as sent by $\mathbf{V}^*$ in the active session of $(i, j, \texttt{msg})$, until the current main message. Now, W^* is ready to use $P^{(i,j,k)}$ to check whether or not this main message and authenticator module from $\mathbf{V}^*$ will be accepted by the active session of $(i, j, \texttt{msg})$.

Before showing how this is done by W^*, we first claim that $P^{(i,j,k)}$'s response to the current main message and the corresponding authenticator is *identical* to the way P would have responded in the active session of $(i, j, \texttt{msg})$. First, note that $\texttt{msg}$ is a valid determining message for the session with $P^{(i,j,k)}$ (even though $\mathbf{V}^*$ sent it in a session with $P^{(i,j,k')}$ for some $k' \neq k$). This is because the validity of $\texttt{msg}$ can depend only on the prover initialization message, which in turn depends only on the input x_i and the random string $\omega_j^{(1)}$. Since x_i and $\omega_j^{(1)}$ are common to both $P^{(i,j,k)}$ and the active session of $(i, j, \texttt{msg})$, we have that the prover initialization message is the same in both cases. Next, we argue that a main message and authenticator module is accepted by the $P^{(i,j,k)}$ *if and only if* it is accepted by the active session of $(i, j, \texttt{msg})$. This is because $P^{(i,j,k)}$ and the active session of $(i, j, \texttt{msg})$ use the same randomness (i.e., $\omega_j^{(1)}$) for verifying the authenticator modules. Furthermore, the authentication procedure of P depends only on this randomness, the verifier's determining message and the current main message. Therefore, if these authenticator modules are accepted by the active session of $(i, j, \texttt{msg})$, then they are also accepted by $P^{(i,j,k)}$, and visa versa. This implies that all the main messages and authenticator modules sent by W^* to $P^{(i,j,k)}$ are accepted by $P^{(i,j,k)}$ (recall that these authenticator modules were previously accepted by the active session of $(i, j, \texttt{msg})$). We therefore conclude that at this point $P^{(i,j,k)}$ expects to receive a main message from V^* (and has not aborted), exactly as P in the active session of $(i, j, \texttt{msg})$. Now, the argument above also implies that if $P^{(i,j,k)}$ accepts the current main message and authenticator module from $\mathbf{V}^*$, then P in the active session of $(i, j, \texttt{msg})$ will also accept the current main message authenticator module.

Thus, all W^* needs to do in order to check if the current main message and authenticator module will be accepted by the active session of $(i, j, \texttt{msg})$ is to see if $P^{(i,j,k)}$ accepts them. So,

[20] One may think that a possible solution to this problem is to first have W^* verify the authenticator module itself. Then, if the authentication succeeds, W^* knows that the main message is valid. Therefore, W^* can forward this message to P and continue with the authenticator module (after "rewinding" $\mathbf{V}^*$ back to this point). The problem that arises with such a strategy is that although W^* may accept the authenticator, this does not mean that P will. (For example, $\mathbf{V}^*$'s strategy may be such that the authenticator is accepted with probability 1/2. Then, even though the main message is valid, W^* may accept the authenticator and P may reject it.) Furthermore, by the definition of admissible protocols, we allow the authenticator module to be such that only P (with its randomness $\omega^{(1)}$) can authenticate it. This means that W^* must use P in some way in order to verify the module, *without* aborting an *active* session in the case that the authentication fails.

W^* continues by sending the current main message to $P^{(i,j,k)}$ and forwarding the messages of the authenticator module between $\mathbf{V}^*$ and $P^{(i,j,k)}$. Following this, W^* receives $P^{(i,j,k)}$'s response (which is either `abort` or the next prover message). We differentiate between two cases:

- If $P^{(i,j,k)}$ aborts after completing the authentication, then W^* sends the standard `abort` message to $\mathbf{V}^*$. We stress that in this case no messages are sent to the active session of $(i, j, \texttt{msg})$.
- If $P^{(i,j,k)}$ does not abort after completing the authentication, then we know that the main message is valid *and* the accompanying authenticator module will be accepted in the active session of $(i, j, \texttt{msg})$. We now distinguish two cases, depending on whether this is the first time that an accepting main-message and authenticator of the current round was sent in a session of $\mathbf{V}^*$ with $\mathbf{P}^{(i,j)}$, in which the determining message equals `msg`. Let $\xi > 1$ denote the index (within the current session) of the current message sent by $\mathbf{V}^*$.
  (a) *The current session is the first session of* $\mathbf{V}^*$ *with* $\mathbf{P}^{(i,j)}$ *in which the determining message equals* `msg` *and the* ξ^{th} *verifier-message along with its authenticator module is accepted by some* $P^{(i,j,\cdot)}$: In this case W^* forwards the current main message to the active session of $(i, j, \texttt{msg})$. Then, W^* rewinds $\mathbf{V}^*$ to the point after it sends the main message (before beginning the authenticator module) and relays all messages from the authenticator module between $\mathbf{V}^*$ and the active session of $(i, j, \texttt{msg})$. Finally, after completing the authenticator module, W^* obtains P's response (i.e., its response to the main message), stores it, and forwards it to $\mathbf{V}^*$.
  (As explained above, we are assured here that W^* obtains the next prover message from the active session of $(i, j, \texttt{msg})$ and not `abort`.)
  (b) *The current session is* NOT *the first session of* $\mathbf{V}^*$ *with* $\mathbf{P}^{(i,j)}$ *in which the determining message equals* `msg` *and the* ξ^{th} *verifier-message along with its authenticator is accepted by some* $P^{(i,j,\cdot)}$: In this case W^* does not communicate with any session of P. Instead, it merely retrieves the corresponding prover response from its storage, and forwards it to $\mathbf{V}^*$. (Here, W^* does not rewind $\mathbf{V}^*$ and as such the next message that $\mathbf{V}^*$ expects to receive is indeed the prover's response.) Note that the corresponding answer is stored in the history of the active session of $(i, j, \texttt{msg})$.[21]
  (Note that by Definition 4.1, it is infeasible for $\mathbf{V}^*$ to send two different main messages along with authenticator modules that are accepting in the same round of two sessions starting with the same verifier determing message. Thus, the responses of $\mathbf{P}^{(i,j)}$ to valid ξ^{th} messages, in sessions starting with the same determining message, are identical. It follows that $\mathbf{V}^*$ will be content with the identical responses supplied to it by W^*.)

5. $\mathbf{V}^*$ *terminates*: When $\mathbf{V}^*$ sends a termination message, which includes its output, W^* just outputs this message and halts.

We stress that W^* is defined to operate in the hybrid model. That is, in every session it invokes with P, a different incarnation is used, and furthermore for every k the adversary W^* holds at most one session with an incarnation of the form $P^{(\cdot,\cdot,k)}$. Thus the second part of P's random-tape in this

[21] We stress that it is also imperative in this case that W^* first verifies that P will accept the authenticator. This is because it is possible that in the past V^* sent the same main message with an accepting authenticator and this time V^* sends it with a rejecting authenticator. Therefore, were W^* to reply to V^* with the next prover message without first checking if P accepts the authenticator, the simulation would no longer be accurate.

session is independent from the random-tape in all other sessions. In contrast, $\mathbf{V}^*$ that operates in the (stronger) resettable model may invoke each incarnation of $\mathbf{P}$ many times, and so the tape $\omega^{(2)}$ as determined (by the same incarnation of $\mathbf{P}$) in these sessions is identical. Nevertheless, we claim that the output of W^* is computationally indistinguishable from the output of $\mathbf{V}^*$. The key observations justifying this claim refer to the actions of $\mathbf{P}$ in the various sessions invoked by $\mathbf{V}^*$:

- In sessions having different determining messages, the second parts of the random-tape (i.e., the $\omega^{(2)}$ part) are independent. The same is true for sessions in which a different incarnation $\mathbf{P}^{(i,j)}$ is used. This is because $\mathbf{P}$ determines $\omega^{(2)}$ by applying a random function on the triplet $(x_i, \omega_j^{(1)}, \texttt{msg})$, where $\texttt{msg}$ is the determining message. (Recall that for $i \neq i'$, the inputs x_i and $x_{i'}$ are distinct and thus for different incarnations of $\mathbf{P}$, the randomness of $\omega^{(2)}$ is independent.) This explains the strategy of W^* in opening a new incarnation of $P^{(i,j,k)}$ whenever V^* sends a new determining message (or invokes a new $\mathbf{P}^{(i,j)}$).
- In sessions having the same common-input, the same $\omega^{(1)}$, and the same determining message, the actions of $\mathbf{P}$ are essentially determined by the determining message. This is because in this case $\mathbf{P}$ determines the same $\omega^{(2)}$, and practically the only freedom of $\mathbf{V}^*$ with respect to the main messages is to choose whether to send the predetermined value or to abort. Thus, $\mathbf{P}_2$'s responses (that are dependent only on $\omega^{(2)}$ and $\mathbf{V}^*$'s main messages) are also determined here. We note that by the definition of admissible protocols, $\mathbf{V}^*$'s lack of freedom in sending its main messages holds even when $\mathbf{V}^*$ can execute a resetting attack (and it therefore holds here, in the hybrid setting, in which $\mathbf{V}^*$ is emulated).

  Therefore, the transcripts of all these sessions correspond to various (augmented) prefixes of one predetermined transcript, where each prefix is either the complete transcript or a strict prefix of it augmented by an `abort` message. This explains W^*'s strategy of replaying the prover's response in the case that V^* repeats a main message in a session with the same determining message.

The corresponding transcripts (of imaginary sessions with $\mathbf{P}$) are generated by W^* by merely copying from real sessions it conducts with P. Each set of $\mathbf{P}^{(i,j)}$-sessions sharing the same determining message, is generated from a single (distinct) session with P (called the active session of that message). The way in which W^* detects and handles invalid main messages of $\mathbf{V}^*$ (or likewise, main messages with accompanying authenticator modules that fail) guarantees that it never aborts an active session, and so such a session can always be extended (up-to completion) to allow the generation of all $\mathbf{P}^{(i,j)}$-sessions sharing that determining message. We stress again that W^* does not need to (and in fact does not) abort a session in order to produce $\mathbf{P}$'s abort message; it merely determines whether $\mathbf{P}$ aborts (and, if so, generates the standard `abort` message by itself).

Finally, we note that if the original protocol (P, V) is a system of arguments *of knowledge*, then so too is the transformed protocol $(\mathbf{P}, \mathbf{V})$. This is because the extractor (of (P, V)) must work for any P^*, and in particular for a P^* that works as defined in the transformation. (Recall that $\mathbf{V}$ is identical to V.) ■

4.4 Discussion

The above proof of the transformation of hybrid protocols to resettable ones has some essential differences to the analogous proof in [7]. In particular, unlike here, the authenticator of an admissible protocol as defined in [7] is universally verifiable. Thus, in the emulation by W^*, machine W^* is able to verify *itself* whether or not P will accept a message from $\mathbf{V}^*$. This also means that for every session with $\mathbf{P}$ opened by $\mathbf{V}^*$, machine W^* opens a single session with P. This is very different from what we have done above, where W^* used P in order to verify authenticators (because it cannot authenticate messages itself). Thus, W^* opens a new incarnation of $P^{(i,j,\cdot)}$ for *every main message* sent by $\mathbf{V}^*$.

The motivation regarding the use of the hybrid model is also very different. In [7], the hybrid model was introduced in order to overcome a technical difficulty; specifically, that known proof systems start with the prover sending a message (this message is the receiver-message of a two-round perfectly-hiding commitment scheme). However, the intuition underlying the transformation (obtaining resettability) is to have the verifier send the first message that then determines all its future messages. The hybrid model was thus introduced in order to allow a separate analysis of the setting in which (only) the first prover message may be re-used (or "reset"). On the other hand, our use of the hybrid model enables us to analyze the case that the prover may be reset during the authentication of every main message (and where this authentication may be interactive and dependent on the prover's re-useable randomness).

5 rZK and rWI Arguments of Knowledge

In this section we will prove Theorem 1.2. We begin by proving (in Section 5.1) the first part of the theorem; that is, the existence of constant-round rWI arguments of knowledge for $\mathcal{NP}$. We do this by constructing an admissible hWI argument of knowledge for $\mathcal{NP}$ and then applying the transformation of Construction 4.3. Next, we prove the second part of the theorem; that is, the existence of rZK arguments of knowledge. We present two alternative constructions of rZK arguments of knowledge (both using the construction of Section 5.1 as a building block):

1. The first construction (presented in Section 5.2) involves a modification of the Richardson-Kilian protocol [32] while using an (admissible) rWI argument of knowledge.
2. The second construction (presented in Section 5.3) works by combining any rWI argument of knowledge, any rZK proof and any perfectly-binding commitment scheme. We note that rZK proofs are known to exist by [7].

In both cases, improved (i.e., poly-logarithmic) round complexity is obtained by referring to the analysis of [26].

We stress that our arguments of knowledge are according to Definition 2.2, where the extractor is given access to the description of the prover's strategy (and not just black-box access). As we have mentioned, this additional information is essential for obtaining rWI or rZK arguments of knowledge.

5.1 Resettable Witness Indistinguishable Arguments of Knowledge

We now present a rWI argument of knowledge for Hamiltonicity. The core of the protocol is the 3-round proof of Hamiltonicity of Blum (see Appendix). As in [21], we augment Blum's protocol[22] by having the verifier initially commit to its query string (rather than choose it after the prover's commitment). Then, after receiving the prover's commitments, the verifier reveals its query string and "proves" to the prover that this is indeed the string that was committed to in the first step of the protocol. In the protocol of [21], the verifier proves the consistency of the query string

[22]The reason we use Blum's protocol rather than the 3-colorability protocol of [23] is that knowledge-extraction is easier with it. The protocol of [21] was not a proof of knowledge anyhow, and so there was no reason to prefer one protocol over the other.

with the initial commitment by simply decommitting. (The prover then continues as in Blum's protocol.) In contrast, in our protocol (below), the verifier proves the consistency of the query string using a resettably-sound zero-knowledge argument. The zero-knowledge proof used must be resettably-sound in order to protect the prover, who may be reset, against a cheating verifier. (This technique of achieving simulation by proving the validity of the revealed value rather than actually decommitting was introduced in [27].) Since the verifier is bound to its initial commitment by the above argument, the fact that the protocol is hWI is shown in a similar manner to the proof (shown in [7]) that the protocol of [21] is hWI. Thus, using the transformation of Construction 4.3 we obtain a rWI protocol.

Indeed, the novelty of our protocol is that it is an argument *of knowledge* (for Hamiltonicity) rather than merely an argument of membership. That is, we construct an extractor K who, given access to the code of a prover P^*, extracts a Hamiltonian cycle (with probability negligibly close to the probability that P^* convinces V). In general, the strategy for extraction from the basic proof of Hamiltonicity (of Blum) involves obtaining the answer to two *different* query strings with respect to the *same* set of prover commitments. The real verifier is unable to do this since it is bound to its queries by the initial commitment (otherwise, the protocol would clearly not be witness indistinguishable). However, K has an advantage over the verifier in that it has access to the code of P^*. Therefore, K can run the (non black-box) simulator for the resettably-sound zero-knowledge proof that asserts the validity of the decommitment. This enables K to cheat and "decommit" to different query strings, thus extracting the Hamiltonian cycle.

As we have mentioned, we construct only a hWI argument of knowledge here and use the transformation of Construction 4.3 to convert this into a rWI argument of knowledge.

Protocol 5.1 (hWI Argument of Knowledge for Hamiltonicity):

- Common Input: *A directed graph* $G = (V_G, E_G)$, *with* $n \stackrel{\text{def}}{=} |V_G|$.
- Auxiliary Input for P: *a directed Hamiltonian Cycle,* $C \subset E_G$, *in* G.
- Fixed Randomness for P: $\omega = (\omega_1, \omega_2) \in_R \{0,1\}^{2n}$.
- The Protocol:
  1. *V chooses a random query string $q \in_R \{0,1\}^n$ and sends a perfectly-binding commitment $c = \mathrm{Commit}(q) = C(q;r)$ to P.*
  2. *P selects n random permutations $\pi_1, \ldots, \pi_n$ of the vertices V_G and sends the verifier V (perfectly binding) commitments to the adjacency matrices of the resulting permuted graphs. That is, P sends an n-by-n matrix of commitments so that the $(\pi_i(j), \pi_i(k))$'th entry is a commitment to 1 if $(j,k) \in E$ and is a commitment to 0 otherwise. Denote by c_i the commitment to the adjacency matrix of $\pi_i(G)$.*

     All the random choices in this step are determined by ω_2.
  3. *V sends q to P* (without decommitting).
  4. *V proves to P that there exists a pair (q,r) such that $c = C(q;r)$, using a* resettably-sound *zero-knowledge proof system (i.e., V proves that q is the correct decommitment).*

     All the random choices of P in this step are determined by ω_1.
  5. *If P accepts the proof (in Step 4), then it replies to the queries q_i (for every $1 \leq i \leq n$) as follows.*
     - *If $q_i = 0$, then P sends π_i along with all the decommitments to c_i.*

- *If* $q_i = 1$*, then* P *decommits to the* n *entries* $(\pi_i(j), \pi_i(k))$ *of* c_i *with* $(j,k) \in C$*.*

6. *For every* i*,* $1 \leq i \leq n$*,* V *checks* P*'s replies as follows.*
   - *If* $q_i = 0$*, then* V *checks that the revealed graph is isomorphic to* G *(with isomorphism* π_i*).*
   - *If* $q_i = 1$*, then* V *checks that all* n *revealed values equal* 1 *and that the corresponding entries form a simple* n*-cycle.*

   In both cases V *also checks that the prover's decommitments are proper.*

   V *accepts if and only if all the above conditions hold.*

Theorem 5.2 *Protocol 5.1 is an admissible system of arguments of knowledge that is* hWI*.*

Proof: We begin by showing that the protocol is sound as an argument of knowledge (with respect to computationally bounded provers). (Completeness is trivial.)

Lemma 5.1.1 (knowledge soundness with negligible error): *There exists a probabilistic expected polynomial-time knowledge algorithm (i.e., knowledge extractor)* K *such that for every probabilistic polynomial-time* P^**, every polynomial* $p(\cdot)$ *and every sufficiently large graph* G*,*

$$\Pr[K(\mathsf{desc}(P^*), G) \in \mathrm{HAM}(G)] > \Pr[\langle P^*, V\rangle(G) = \text{ACCEPT}] - \frac{1}{p(|G|)}$$

where $\mathsf{desc}(P^*)$ *denotes the description of* P^**'s strategy and* $\mathrm{HAM}(G)$ *denotes the set of Hamiltonian cycles in* G*.*

Proof: The principle upon which the extraction is based is that if K obtains replies from P^* to two different query strings (for the same series of permuted adjacency matrices), then the Hamiltonian cycle can be retrieved. This is because for some i, the extractor K obtains $\pi_i(G)$,π_i and a simple cycle of length n in $\pi_i(G)$. Therefore, it is easy for K to compute a simple cycle of length n in G itself.

However, notice that K is committed to its query string *before* the prover commits to the adjacency matrices. Thus, we must show how, despite this initial commitment by K, it is able to obtain replies to different query strings. The key point is that K can cheat in the zero-knowledge proof of Step 4 by running the proof simulator. Then, due to the hiding property of the commitment scheme, P^* cannot distinguish this behavior from the behavior of an honest verifier. Thus, K can "decommit" in two different ways and obtain replies for different queries. As discussed above, this enables K to reconstruct a Hamiltonian cycle in G.

The extractor K (upon input a description of P^* and the graph G) works as follows:[23]

1. K chooses a uniform string R for the random tape of P^*.

[23] The extraction procedure described here is slightly non-standard in the following sense. Usually, extraction works by K first verifying the proof from P^* (according to the instructions of an honest verifier). Then, if the proof is accepting, it begins a procedure that involves many rewindings until enough information is gathered to construct a witness. Furthermore, this second stage usually takes time that is inversely proportional to the probability that the verification of the first stage succeeds. Thus, the overall running time of K remains expected polynomial-time. However, were we to use such a strategy, our analysis would become considerably more complex. In particular, in our case the second stage would take time that is inversely proportional to a function that is *negligibly close* (and not equal) to the probability that the verification of the first stage succeeds. Thus, in order to ensure expected polynomial-time, we would need to apply the (somewhat complex) techniques of [21].

2. K chooses $q \in_R \{0,1\}^n$, computes $c = \text{Commit}(q) = C(q;r)$ for a uniform r and (internally) passes c to P^*. K then receives a message from P^* which is denoted by $c_1, \ldots, c_n$. Formally, this is obtained by K computing $P^*(G,R,c)$, where $P^*(G,R,\overline{m})$ denotes the message sent by P^* upon input G, random tape R and sequence of incoming messages $\overline{m}$.

3. K chooses $q' \in_R \{0,1\}^n$ (independently of the above q) and sends q' to P^*. Furthermore, K runs the zero-knowledge simulator for the proof given to P^* in Step 4, where the verifier is defined by $P^*(G,R,c,q')$. Denote the resulting transcript by t_{pf}. (Note that since the simulator for the zero-knowledge proof must be non black-box, K needs non black-box access to P^* to implement this stage.[24])

4. K obtains P^*'s decommitments by computing $P^*(G,R,c,q',t_{pf})$.

5. K verifies the answer according to V's instructions.

   (a) If the verification fails, then K halts with output $\perp$.

   (b) If the verification succeeds, then K continues by repeating Steps 3–5 until another success occurs. Denote the string for which the success occurs by q''. (In each of these repetitions, K uses the same randomness for the zero-knowledge simulation; this technical point simplifies the analysis.)

   If $q'' \neq q'$, then extraction succeeds and K outputs the cycle C. On the other hand, if $q'' = q'$, then extraction fails and K outputs $\mathsf{failure}$.

We first claim that the probability that K halts with output $\perp$ is negligibly close to the probability that V rejects a proof upon interaction with P^*. That is:

Claim 5.1.2 *For every probabilistic polynomial-time prover P^*, every polynomial $p(\cdot)$ and every sufficiently large graph G,*

$$|\Pr[\langle P^*, V\rangle(G) = \text{REJECT}] - \Pr[K(\mathsf{desc}(P^*), G) = \perp]| < \frac{1}{p(|G|)}$$

Proof: Notice the following two differences between a transcript generated by K (in Steps 1–4) and the transcript of a real (P^*, V) execution:

- The query string q' sent by K is independent of the value initially committed to.
- The zero-knowledge proof is simulated and not real.

Intuitively, despite these differences, the distribution of transcripts generated by K is computationally indistinguishable from transcripts resulting from real (P^*, V) executions. This is due to the hiding property of the commitment scheme and the indistinguishability of simulated proofs from real ones. Now, Protocol 5.1 is such that given the proof transcript one can easily derive the verifier's decision of whether to accept or reject the proof from P^*. Thus the indistinguishability of the transcripts implies that the probability that K outputs $\perp$ (which occurs when P^* fails to supply a convincing proof) is negligibly close to the probability that V rejects after interaction with P^*. (Otherwise, this could be used to distinguish the transcripts.) We now formally prove that the

[24] Recall that the proof here is resettably-sound and therefore by Theorem 3.4, the simulator here must be non black-box.

transcripts in the above two scenarios are indeed indistinguishable. This proof is quite standard and may be skipped with little loss.

First consider a hybrid, mental experiment in which the query string sent by K is indeed the same string as that committed to initially, yet the zero-knowledge proof is simulated. We claim that the transcript output in this hybrid experiment is computationally indistinguishable from the transcript of a (P^*, V) execution. This is because the only difference between them is that in the transcript output in the hybrid experiment, the zero-knowledge proof is simulated and not a real proof. Therefore, indistinguishability is guaranteed by the zero-knowledge property of the proof; specifically by the formulation of zero-knowledge with respect to auxiliary inputs.

Next, we claim that the transcript output in the hybrid experiment is indistinguishable from a transcript output by K (where the only difference is regarding the value of the initial commitment). Intuitively, this is due to the hiding property of the commitment scheme. Formally, if the transcripts can be distinguished, then we can construct a distinguisher D for the commitment scheme in the following sense. D generates two independent and uniformly distributed strings of length n: q_0 and q_1. Next, a bit $b \in_R \{0,1\}$ is chosen and D receives a commitment c, where $c = \text{Commit}(q_b)$, and D's goal is to attempt to guess b with probability non-negligibly greater than 1/2. Indeed, D succeeds in its goal by simulating an interaction with P^* as follows (D internally incorporates P^* and works similarly to K). First D (internally) passes P^* its challenge commitment c. Then, after receiving P^*'s reply (which should consist of commitments to adjacency matrices), it passes P^* the query string q_0 and runs the simulator for the zero-knowledge proof of Step 4 with P^* as the verifier. Finally, D runs a distinguisher D' on the resulting transcript, outputting whatever D' does. Now, consider the transcript resulting from D's simulation. If $b = 0$ (i.e., $c = \text{Commit}(q_0)$), then the transcript has exactly the same distribution as a transcript from the hybrid experiment. On the other hand, if $b = 1$ (i.e., $c = \text{Commit}(q_1)$), then the transcript has exactly the same distribution as a transcript output by K. We conclude that if there exists a D' who, with non-negligible probability, can distinguish transcripts from the hybrid experiment from transcripts output by K, then D can distinguish the commitments with non-negligible probability. By combining the above, we conclude that the distribution of transcripts output by K is computationally indistinguishable from the distribution of transcripts resulting from (P^*, V) executions. ■

Having bounded the probability that K outputs $\perp$, we now bound the probability that it fails. That is,

Claim 5.1.3 *For every probabilistic polynomial-time prover P^* and every sufficiently large graph G (over n nodes),*

$$\Pr[K(\mathsf{desc}(P^*), G) = \mathsf{failure}] = \frac{1}{2^n}$$

Proof: Recall that K outputs failure in the event that it reaches Step 5 *and* $q'' = q'$. Denote by R' the (fixed) coins used by K in the simulation of the zero-knowledge proof given to P^* in Step 4 and denote by $p_{R,R'}$ the probability (over K's other coin tosses) that the proof (of Steps 1–4) from P^* is successful, when P^* uses the random string R. (We say that the proof is successful if the verification by K succeeds.) Thus, the probability that K reaches Step 5(b) equals $p_{R,R'}$ exactly. Now, since K uses the same random coins in the simulated proof each time (i.e., R' is fixed), the probability that P^* (with fixed randomness R) successfully completes the proof is dependent on the choice of q only (and not on the zero-knowledge simulation). This means that there are exactly $p_{R,R'} \cdot 2^n$ different strings q for which P^* successfully completes the proof. Therefore, given that

K reaches Step 5, the probability that the same success string is obtained equals $\frac{1}{p_{R,R'} \cdot 2^n}$. We conclude that the probability that K outputs $\mathsf{failure}$ equals $\frac{p_{R,R'}}{p_{R,R'} \cdot 2^n} = 2^{-n}$. ■

We have shown that K outputs $\mathsf{failure}$ with probability 2^{-n}. Furthermore, we have shown that the probability that K outputs $\perp$ is negligibly close to the probability that V rejects. Finally, notice that if K does not output $\mathsf{failure}$ and does not output $\perp$, then this means that it outputs a Hamiltonian cycle. Therefore, by combining Claims 5.1.2 and 5.1.3, we have that the probability that K extracts a Hamiltonian cycle is negligibly close to the probability that V accepts in an interaction with P^*.

It remains to show that K runs in expected polynomial-time. Recall that the extraction works by running the protocol (with the simulated proof and independent query string) once and then in case the first execution was accepting, this is repeated until another success occurs. Now, each repetition of Steps 3–5 involves a polynomial number of steps plus a single invocation of the (resettably-sound) zero-knowledge simulator. Recall that the random coins R' used by the simulator are fixed and that the running-time depends only on this R'. (We note that the running-time does *not* depend on P^*'s random coins R. This is because they are previously fixed and the simulator must work for any verifier, in particular a verifier with fixed random coins.) We thus denote by $t_{R'}$ the (exact) running-time of the simulator where the simulator's coins R' are fixed. Therefore, the expected running-time of the extractor is:

$$\frac{1}{2^{|R'|}} \sum_{R'} \left(1 + p_{R,R'} \cdot \frac{1}{p_{R,R'}}\right) (\mathrm{poly}(n) + t_{R'}) = \mathrm{poly}(n) + \frac{1}{2^{|R'|}} \sum_{R'} t_{R'}$$

where $p_{R,R'}$ equals the probability (over K's coin tosses with the coins of the simulator fixed by R') that the verification of the proof received by K from P^* is successful and the $\mathrm{poly}(n)$ factor includes all steps of K excluding the simulation of the zero-knowledge proof. Since the simulator is expected polynomial-time, we have that the overall expected running-time of the extractor is polynomial, as required. ■

Having established that Protocol 5.1 is a system of arguments of knowledge, we proceed to show that it is hWI.

Lemma 5.1.4 (hybrid witness indistinguishability): *Protocol 5.1 is an admissible system of arguments that is* hWI.

Proof: It is easy to verify that Protocol 5.1 is admissible: the first message sent by the verifier (i.e., in Step 1) is a commitment to its query string that is later sent in Step 3 (this is the determining message). There is only one main message sent by V (i.e., in Step 3) and one accompanying authenticator module (Step 4). By the resettable-soundness of the zero-knowledge argument of Step 4, we have that except with negligible probability, in two executions with the same determining message, V cannot send two different messages in Step 3 such that P accepts both. (We stress that due to the resettable soundness of the zero-knowledge argument, this holds even when V can execute a resetting-attack on P.) Finally, we note that syntantically P fulfills the requirements of an admissible protocol. That is, P indeed consists of two parts, where P_1 controls Steps 4 and 5 and uses randomness ω_1, and P_2 controls Steps 2 and 5 and uses randomness ω_2. Furthermore, P_2 computes its response (of Step 5) based only on the main message sent by V in Step 3.

Next, notice that Protocol 5.1 is *zero-knowledge* in a *standard setting.* This can be shown using almost the same argument as in [21]. (In fact the only difference is that in [21] the verifier decommits to its query string, whereas here it proves that its decommitment is correct. Thus, there is at most a negligible difference.) Thus, it follows that Protocol 5.1 is *witness indistinguishable* in the *concurrent setting* (cf. [13]). However, we need to show that it is witness indistinguishable in the *hybrid model*. Denote the randomness used by P in generating the commitments in Step 2 by ω_2 and the randomness used by P in verifying the zero-knowledge proof of Step 4 by ω_1. Then, the extra power of the adversary in the hybrid model is to invoke sessions with the same (random) value of ω_1. (Recall that in the hybrid model, V^* may invoke different incarnations of P, using the same randomness for the prover initialization message and for the verification of the authenticator modules.) However, P only uses the random string ω_1 for verifying a *resettably-sound* proof. Therefore, soundness holds even when ω_1 is reused in a polynomial number of different sessions. (Note that here we need the adaptive version of soundness where the statements are not chosen by the adversary ahead of time, but are rather selected on the fly.) This means that if we limit V^*'s view to the messages outside of this zero-knowledge argument, then there is at most a negligible difference between V^*'s view in the concurrent and hybrid models. Therefore, Protocol 5.1 remains witness indistinguishable in the hybrid model. This completes the proof. ■

Combining Lemmas 5.1.1 and 5.1.4, we complete the proof of Theorem 5.2. ■

By combining Theorem 4.4 with Theorem 5.2 we obtain the following Corollary:

Corollary 5.3 *The result of applying the transformation of Construction 4.3 to Protocol 5.1 is a system of resettable witness-indistinguishable arguments of knowledge.*

5.2 Deriving a rZK arguments of knowledge for $\mathcal{NP}$

Part 2 of Theorem 1.2 is proved by applying Construction 4.3 to an admissible (as per Definition 4.1) argument of knowledge for $\mathcal{NP}$ that is zero-knowledge in the hybrid model (of Definition 4.2). Thus, we need to assert the existence of such a protocol.

Proposition 5.4 *Suppose that there exists a* two-round *perfectly-hiding commitment scheme. Then, every NP-relation has an admissible argument of knowledge that is zero-knowledge in the hybrid model. Furthermore, the round complexity of this protocol is poly-logarithmic.*

Combining Theorem 4.4 and Proposition 5.4, Part 2 of Theorem 1.2 follows.

Proof Sketch: We show that the concurrent zero-knowledge proof system of Richardson and Kilian [32] (with as setting of parameters as in [26]) can be *modified* so that the resulting protocol satisfies all requirements. Recall that the Richardson and Kilian protocol, hereafter referred to as the RK-protocol, consists of two stages with Stage 2 being any witness-indistinguishable proof (or argument) system applied to a statement defined by Stage 1. The *modification* is analogous to the one presented in [7]: As in [7], we use an admissible witness-indistinguishable (WI) protocol in Stage 2, and the modification amounts to moving the first (i.e., determining) message verifier of the WI protocol to the very beginning of the entire protocol.A key difference (w.r.t [7]) is that we use our (constant-round) witness-indistinguishable argument of knowledge (i.e., Protocol 5.1), rather than using the witness-indistinguishable proof of [21] (or its admissible version derived in [7]). We stress that in contrast to the protocol of [21], which is unlikely to be an argument of knowledge, Protocol 5.1 is an argument of knowledge. Furthermore, like the protocol of [21], Protocol 5.1 is admissible and is witness-indistinguishable in the hybrid model. Thus, the proposition follows by verifying the following:

1. The modified protocol is an argument of knowledge (for the same relation as the original one).

   The key point is to notice that *before modification* the above protocol is an argument of knowledge (and not merely an argument for membership in the language). This fact follows from the soundness proof of the RK-protocol which guarantees that, with overwhelming probability, in real executions the only possible NP-witness for the statement proven (by the WI argument of knowledge) in Stage 2 is an NP-witness for the common input. Thus, using the knowledge-extractor for the WI argument of knowledge (executed in Stage 2), we obtained the desired NP-witness.

   To show that the modified protocol remains an argument of knowledge we observe that the prover is unlikely to gain anything by the modification. This is the case because the message that we are discussing is a commitment (which anyhow yields no knowledge to the prover, and this fact is used anyhow in the soundness proof of the RK-protocol).

2. The modified protocol is admissible and is zero-knowledge in the hybrid model.

   Recall that the verifier's first message in the modified protocol consists of two strings, the first being the first message of Stage 1 and second being the first message of Stage 2 in the original (unmodified protocol). The determining feature of the first string (w.r.t Stage 1) is proven exactly as in [7], whereas the determining feature of the second string (w.r.t Stage 2) is proven as in Lemma 5.1.4. The modified protocol is clearly concurrent zero-knowledge (because we have only weakened the cheating verifier by the modification), and the fact that it is zero-knowledge in the hybrid model is proven analogously to the proof in [7].

The proposition follows. ■

5.3 rZK arguments of knowledge for $\mathcal{NP}$: an alternative construction

In this section, we present an alternative construction of a rZK argument of knowledge. Our construction uses a rWI argument of knowledge (as presented in the previous section), a rZK argument system (these exist as was shown in [7]) and a perfectly-binding commitment scheme. A basic outline of the protocol is as follows (let x be the statement to be proved and w a witness for the fact that $x \in L$):

1. The prover commits to w using a perfectly-binding commitment scheme.

2. The provers proves in rZK that the above commitment is correct. That is, it proves that the value committed to in Step 1 is indeed a witness proving that $x \in L$.

3. The prover proves that it knows the value committed to in Step 1 using a rWI argument of knowledge.

Loosely speaking, zero-knowledge is demonstrated by the following simulation strategy. The simulator first commits to garbage, then runs the simulator for the rZK of Step 2, and finally proves that it knows the decommitment (i.e., executed in Step 3). Using the indistinguishability of commitments, and the indistinguishability of real Step 2 proofs from simulated ones, we show that such a simulator generates transcripts that are indistinguishable from real executions of the entire protocol. (The actual simulation is more complicated as it must work for a verifier that can carry out a resetting attack on the prover.) On the other hand, extraction is carried out via the rWI argument of knowledge, and the rZK proof guarantees that the extrcated string is indeed a witness.

Before providing a detailed description of the protocol and its proof, we note that we actually need something stronger than witness indistinguishability in Step 3 of our construction. Intuitively, the simulator described above generates good (i.e., indistinguishable from real) transcripts, as long as the proof it emulates for Step 3 is indistinguishable from the proof given in a real execution of Step 3 (in a real proof where the commitment is to a valid witness). But witness indistinguishability does not guarantee the latter, becuase it only relates to possible witnesses (i.e., decommitments values) to a fixed statement (i.e., a commitment), and in this case there is a *single* witness (i.e., decommitment value). What we need is a stronger notion (than WI) called *strong witness indistinguishable*, which guarantees that if the commitments are indistinguishable before the proof then they remain indistinguishable even after seeing the proof. Indeed, in our case, a commitment to garbage is indistinguishable from a commitment to a real witness (for $x \in L$), and therefore strong-WI implies that the proof emulated (by our simulator) for Step 3 is indistinguishable from the proof given in a real execution of Step 3.

Definition 5.5 (strong witness indistinguishability [19]): *Let (P, V) be an interactive proof for an NP-language L, and let R_L be a witness relation for the language L. We say that (P, V) is* strong witness indistinguishable for R_L *if for every probabilistic polynomial-time interactive machine V^* and every two probability ensembles $\{(X_n^1, Y_n^1, Z_n^1)\}_{n\in\mathbb{N}}$ and $\{(X_n^2, Y_n^2, Z_n^2)\}_{n\in\mathbb{N}}$ so that each (X_n^i, Y_n^i, Z_n^i) ranges over $(R_L \times \{0,1\}^*) \cap (\{0,1\}^n \times \{0,1\}^* \times \{0,1\}^*)$, the following holds:*

> *If $\{(X_n^1, Z_n^1)\}_{n\in\mathbb{N}}$ and $\{(X_n^2, Z_n^2)\}_{n\in\mathbb{N}}$ are computationally indistinguishable, then so are $\{\langle P(Y_n^1), V^*(Z_n^1)\rangle(X_n^1)\}_{n\in\mathbb{N}}$ and $\{\langle P(Y_n^2), V^*(Z_n^2)\rangle(X_n^2)\}_{n\in\mathbb{N}}$.*

Thus, by defining X_n^1 to be the distribution over commitments to valid witnesses (followed by a simulated rZK transcript) and X_n^2 the distribution over commitments to garbage (followed by a simulated rZK transcript), we have that after a strong witness indistinguishable proof, these distributions are still indistinguishable. Therefore, the simulator generates transcripts that are indistinguishable from in a real execution.

Before proceeding, we must show the existence of resettable *strong* witness indistinguishable (denoted rSWI) arguments of knowledge. However, we claim that applying the transformation of Construction 4.3 to Protocol 5.1 yields a resettable strong WI system of arguments of knowledge. This is obtained by combining Theorem 4.4 with the following proposition.

Proposition 5.6 *Protocol 5.1 is an admissible system of arguments of knowledge that is* strong witness indistinguishable *in the hybrid model.*

The proof of this proposition is identical to the proof of Theorem 5.2. (Note that like standard witness indistinguishability, zero-knowledge implies strong WI, and strong WI remains strong WI under concurrent composition.)

We are now ready to present the protocol for an arbitrary NP-relation R_L (corresponding to $L \in \mathcal{NP}$):

Protocol 5.7 (rZK Argument of Knowledge):

- Common Input: $x \in L$
- Auxiliary Input for P: w *such that* $(x, w) \in R_L$
- Fixed Randomness for P: $\omega = (\omega_1, \omega_2, \omega_3) \in_R \{0,1\}^{3n}$.
- The Protocol:

1. *P sends a perfectly-binding commitment $c = \text{Commit}(w) = C(w;r)$ to the verifier V. The randomness r used by P in computing the commitment is obtained by applying a pseudorandom function, keyed by ω_1, to the statement x.*
2. *Using a* rZK *argument, P proves that there exists a pair (w,r) such that $c = C(w;r)$* and *$(x,w) \in R_L$. (That is, P proves in* rZK *that the commitment in Step 1 is correct.)*
   The randomness used by P in proving this proof equals ω_2.
3. *Using a resettable* strong *witness indistinguishable* (rSWI) *argument of knowledge, P proves that it knows a pair (w,r) such that $c = C(w;r)$. (That is, P proves that he knows the decommitment to c.)*
   The randomness used by P in proving this proof equals ω_3.
4. *V accepts if it accepts both the proofs in Steps 2 and 3.*

Theorem 5.8 *Protocol 5.7 is a resettable zero-knowledge argument of knowledge for L.*

Proof: We first show that Protocol 5.7 is an argument of knowledge for R_L. Completeness (or non-triviality) follows by the completeness of the arguments used in Steps (2) and (3). We now show that the soundness (or validity) requirement of arguments of knowledge holds. That is, we show that there exists a expected probabilistic polynomial-time knowledge extractor K such that for every polynomial-size P^* and all sufficiently large x's

$$\Pr[K(\mathsf{desc}(P^*), x) \in R_L(x)] > \Pr[\langle P^*, V\rangle(x) = \text{ACCEPT}] - \frac{1}{poly(|x|)} \tag{1}$$

Intuitively, the extractor works by first verifying the rZK proof, next running the extractor for the rSWI argument of knowledge, and finally outputting the extracted witness. Formally, (on input $\mathsf{desc}(P^*)$ and x) the extractor K works as follows:

- K chooses a uniform string R for the random tape of P^*.
- K invokes P^* on input x and random-tape R and obtains a message c (that should be a commitment to a witness).
- K plays the honest verifier for the rZK proof provided by $P^*(x, R)$ in Step 2. Denote the transcript of the proof by t_{pf}. If K rejects the proof, then it outputs $\perp$ and halts. Otherwise, it continues to the next step.
- K invokes the extractor for the rSWI argument of knowledge giving it input $\mathsf{desc}(P^*(x, R, t_{pf}))$ and c (notice that the prover is defined by $P^*(x, R, t_{pf})$). Denote the output of the extractor by (w', r'). Then, K outputs w' and halts.

We first claim that K runs in expected polynomial-time. This follows from the fact that the extraction procedure of the rSWI argument of knowledge runs in expected polynomial-time (and outputs a witness with probability close to the probability that P^* convinces the verifier), and from the fact that all other steps of K can be completed in strict polynomial-time. Next, we claim that K outputs a valid witness (i.e., w' such that $(x, w') \in R_L$) with probability that is negligibly close to the probability that V accepts. This can be seen as follows:

The probability that K accepts the rZK proof (and thus reaches the rSWI argument of knowledge) *equals* the probability that V accepts the rZK proof. Then, by the property of the extractor of the rSWI argument of knowledge, the probability that (w', r') is such that $c = C(w'; r')$ is negligibly close to the probability that V accepts the rSWI proof from P^*. Therefore, the probability that K

outputs w' such that c is a commitment to w' is negligibly close to the probability that V accepts the entire argument system from P^*. Finally we note that by the soundness of the rZK proof of Step 2 (that is properly verified by K), the probability that w' is not such that $(x, w') \in R_L$ is at most negligible. (In this argument we rely on the fact that the commitment scheme is perfectly binding and therefore there is a single w' that can be used for convincing a verifier in both Steps 2 and 3.) Eq. (1) follows.

Resettable Zero-Knowledge: We now show that Protocol 5.7 is rZK. Intuitively, the simulation is carried out as follows. First, the simulator S commits to garbage (rather than to a real witness). Next, S invokes the simulator for the rZK proof of Step 2. Denote this simulator by S_{rZK}, and denote the corresponding prescribed prover (of the rZK proof of Step 2) by P_{rZK}. Finally, S proves that it knows the decommitment using the rSWI proof of Step 3. Intuitively, the simulator for the rZK proof enables S to cheat and commit to garbage without being detected. The actual simulation is more involved than this. The reason is that the simulator S_{rZK} is only guaranteed to simulate a (resetting) verifier that only receives a rZK proof. However, here we wish to use S_{rZK} in order to simulate a (resetting) verifier who also receives related messages from a rSWI proof. (If we were simulating an ordinary (i.e., non-resetting) verifier then it would have been easy to augment the simulation, but here the verifier may reach Step 3 of some session (i.e., the rSWI proof) and then beging a new interaction with the rZK prover of Step 2.) We solve this technicality by defining a verifier V^{**} who incorporates V^*, and emulates the (first and) third steps of Protocol 5.7 by itself. On the other hand, the messages of the rZK proof of Step 2 are relayed by V^{**} between V^* and the *external* prover. We then run the simulator S_{rZK} on this V^{**}, and thus obtain a simulation of Protocol 5.7.

We begin by defining the verifier V^{**} who is a verifier for a rZK proof that takes place in Step 2 of Protocol 5.7. Recall that this proof system is for the following language:

$$L' \stackrel{\text{def}}{=} \{(x, c) \mid \exists (w, r) \text{ s.t. } c = C(w; r) \ \& \ (x, w) \in R_L\}$$

Loosely speaking, V^{**} internally incorporates V^* and emulates Steps 1 and 3 of Protocol 5.7. (We therefore talk of both internal communication with V^* and external communication with the prover.) Specifically, V^{**} receives inputs $\overline{x}' = ((x_1, c_1), \ldots, (x_m, c_m))$ and auxiliary inputs $\overline{z} = (z_1, ..., z_m)$ and works as follows:

- When V^* initiates an interaction with the $(i,j)^{\text{th}}$ incarnation of P (i.e., $P^{(i,j)}$), then V^{**} acts as follows:
  1. If the auxiliary input z_i equals the decommitment of c_i (i.e., $z_i = (w, r)$ such that $c_i = C(w; r)$), then V^{**} continues to the next step. Otherwise, it halts.
  2. *Internally emulate Step 1:* V^{**} internally passes V^* the commitment c_i and (externally) initiates a session with the $(i,j)^{\text{th}}$ incarnation of P_{rZK} (i.e., $P_{\mathrm{rZK}}^{(i,j)}$).
  3. *Relay Step 2 to the prover P_{rZK}:* For every message m sent by V^* belonging to the rZK proof of Step 2, machine V^{**} (externally) forwards m to $P_{\mathrm{rZK}}^{(i,j)}$, and likewise (internally) returns $P_{\mathrm{rZK}}^{(i,j)}$'s response to V^*.
  4. *Internally emulate Step 3:* For every message m sent by V^* belonging to the rSWI argument of knowledge of Step 3, machine V^{**} uses the knowledge of (w, r) in order to (internally) play the prover's role. (Recall that $c_i = C(w; r)$ and therefore V^{**} can execute the prover's role in this rSWI argument of knowledge. V^{**} proves this exactly as an honest prover would and thus uses fixed randomness ω_3 that is uniformly chosen at the outset.)

- When V^* halts, V^{**} outputs whatever V^* does.

The important property of V^{**} is that when V^{**} receives auxiliary inputs $\overline{z}$ such that for every i, the auxiliary input z_i equals the correct decommitment of c_i, then the output of V^{**} (upon interaction with the prover of the rZK proof) *equals* the output of V^* upon interaction with the prover of Protocol 5.7, conditioned on the commitments generated in Step 1 being equal these c_i's. We are now ready to proceed by defining the simulator S for Protocol 5.7.

The simulator S, on input $\overline{x} = x_1, \ldots, x_m$, works as follows:

- For $i = 1, \ldots, m$, define $c_i = \text{Commit}(0^n) = C(0^n; r_i)$ for uniformly chosen r_i's. Then define the sequence of statements $\overline{x}' = (x'_1, ..., x'_m)$ for the rZK proof of Step 2, where for every i, $x'_i = (x_i, c_i)$. (Recall that the rZK proof of Step 2 is based on statements of the form (x, c).)
- Invoke the simulator for the rZK proof on V^{**} with auxiliary input $\overline{z} = (z_1, ..., z_m)$ defined by $z_i = (0^n, r_i)$. That is, invoke $S_{\text{rZK}}(V^{**}(\overline{z}), \overline{x}')$ and output whatever it outputs.

We claim that the output of S is indistinguishable from V^*'s output in a real execution of Protocol 5.7. We first define a hybrid mental experiment as follows. In this hybrid experiment, the simulator S commits to the real witnesses, rather than to 0^n. Otherwise, everything is the same as in the simulation. Thus, the only difference between the hybrid experiment and the simulation is with respect to the value of the commitments $c_1, \ldots, c_m$ and essentially the only difference between the hybrid experiment and a real execution is the use of the simulator for the rZK instead of providing a real proof.

We now show that the hybrid experiment is indistinguishable from the above simulation by S. As we have mentioned, the only difference between the distributions is whether the commitments are to valid witnesses or 0^n. Thus, by the hiding property of the commitment scheme, the sequence of statements on which the rZK proof of Step 2 is being invoked in the two cases are indistinguishable. Next, we claim that we can ignore the rZK proof of Step 2. This is because the same simulator is used in both experiments (and it receives no input regarding the decommitments) and therefore any distinguisher can internally run it. However, we must show that the above are indistinguishable even after the rSWI argument. This follows immediately by the definition of strong witness indistinguishability (the fact that it is resettable is required because this is the attack executed by V^* in the emulation by V^{**}). Thus, we have that the hybrid experiment and simulation are indistinguishable.

Next, we show that the hybrid experiment is indistinguishable from a real interaction. In order to show this, we first replace the pseudorandom function used by the prover in order to generate the commitments with a truly random function. Due to the pseudorandomness of the function, this change is indistinguishable. Next, we note that the random coin tosses used for generating the commitments in the hybrid experiment are randomly chosen. This is identical to the behavior of the (now modified) prover, since all the x_i's are distinct (and the prover applies a random function to the x_i's for the coins). Thus, the only difference between the hybrid experiment and a real interaction is with respect to the simulated rather than real proof. Since S_{rZK} must succeed in generating an indistinguishable transcript for any verifier, it must in particular succeed in generating such a transcript for the specific verifier V^{**}. As we have mentioned above, when V^{**} interacts with a real prover, its output is identically distributed to V^*'s output in an interaction with the prover of Protocol 5.7. Thus, S_{rZK} must output a distribution that is indistinguishable from V^*'s output as described.

Combining the above, we have that S outputs a distribution that is indistinguishable from V^*'s output upon interaction with the prover of Protocol 5.7. ■

6 Resettable Zero-Knowledge in the Public-Key Model

Canetti et al. [7] introduced a weak notion of a public-key setup, in order to obtain constant-round resettable zero-knowledge arguments (a somewhat stronger assumption was independently suggested by Damgård [9]). The only requirement in this public-key model is that all verifiers deposit some public-key in a public file *before* the prover begins any interaction. We stress that there is no requirement whatsoever on the "validity" of the public keys deposited. The use of the public-file is simply to limit the number of different identities that a potential adversary may assume (note that the adversary may try to impersonate any registered user, but it cannot act on behalf of a non-registered user). For a more detailed description of the public-key model, see [7].

Constant-round resettable zero-knowledge arguments were presented in this model in [7, 29]. However, these constructions require a subexponential hardness assumption. In this section, we present a constant-round resettable zero-knowledge argument of knowledge (in the public-key model) under seemingly weaker assumptions. Specifically, our construction requires the existence of one-way functions and resettably-sound zero-knowledge, where the latter follows from the existence of a family of hash functions that are collision-intractable with respect to some specified super-polynomial security function (say $n^{\log\log n}$). Our protocol and its proof are also much simpler than that of [7, 29]. Furthermore, our argument is an argument of knowledge. Thus, we extend the possibility of obtaining resettable zero-knowledge arguments of knowledge to *constant-round* protocols in the public-key model as well.

6.1 Definition

Micali and Reyzin [30] showed that the notion of soundness in the public-key model is delicate and specifically that it is not enough to consider a single execution. In fact, they define four (separate) levels of soundness: one-time soundness, sequential soundness, concurrent soundness and resettable soundness. Each of these notions refers to an adversarial prover's power when interacting with an honest verifier. That is, for sequential soundness, we say that a protocol is sequentially sound if a cheating prover P^* cannot convince the verifier V of a false statement even after a polynomial number of sequential executions. (We stress that P^* can adaptively choose the statements to be proven.) Concurrent soundness is defined similarly for concurrent executions of P^* with V. (Resettable-soundness concurs with Definition 3.1 when adapted to the public-key model.)

The protocol that we present fulfills *sequential soundness* but is probably not concurrently sound.[25] We now proceed by defining sequentially-sound resettable zero-knowledge.

Definition 6.1 (sequential soundness in the public-key model): *A* sequential attack *of a cheating prover P^* on a verifier V* in the public-key model *is defined by the follow two-step random process, indexed by a security parameter n.*

1. *Run the key-generation stage of V to obtain (PK, VK).*
2. *On input 1^n and PK, machine P^* is allowed to initiate* poly(n)*-many interactions with V. The activity of P^* proceeds in rounds. In round i, the machine P^* chooses $x_i \in \{0,1\}^n$ and interacts with V on input SK and x_i. We stress that the actions of P^* in round i may depend on PK (as well as the history of previous rounds).*

[25] We note that, in practice, this means that a verifier must not open more that one session at a time. This is a significant limitation and it means that although resettable zero-knowledge implies concurrent zero-knowledge *with respect to the prover's security*, it does not necessarily imply that the verifier is protected in a concurrent setting. We stress that this problem only arises in the public-key model and is due to the dependence of different sessions induced by a common (verifier) public-key.

Let P and V be some pair of probabilistic polynomial-time interactive machines. We say that (P, V) is a sequentially-sound system of arguments in the public-key model *for a language L, if it is complete*[26] *and for every polynomial-size P^* executing a sequential attack on V, the probability that there exists an i such V accepts in the i'th round and $x_i \notin L$ is negligible.*

Naturally, a system of arguments (in the public-key model) is said to be *sequentially-sound resettable zero-knowledge* if it is both sequentially-sound and resettable zero-knowledge.

6.2 The Protocol

Similarly to the public-key protocol of [7], in the first stage of our protocol the verifier proves a zero-knowledge argument of knowledge of the secret-key associated to its public-key (in the public file). The nature of the public-key is such that simulation is made "easy" (without any necessity for rewinding) if the associated secret-key is known. Therefore, simulation proceeds by first extracting the secret-key and then completing the simulation (without any rewinding). As in [7], the role of the public file is to prevent the adversary from using "too many" public keys. We note, however, that (for reasons to be discussed below) the second stage of our protocol is completely different from the protocol of [7]. Despite this difference, both protocols share the property that the "hard part" of the simulation is extracting the secret-key and the "easy part" is simulating the second stage, given the secret key. Another important difference between our protocol and the protocol of [7] relates to the argument of knowledge used by the verifier in the first stage. We use a resettably-sound zero-knowledge argument of knowledge, whereas they rely on a (resettably-sound) proof system that can be simulated in subexponential time. This difference expresses itself in the proof of soundness, and is one of the reasons why we get away with a weaker intractability assumption (while using a simpler proof of soundness). The other reason is that our protocol avoids issues related to the malleability of commitment schemes (which are resolved in [7] by using a subexponential intractability assumption). We now briefly describe the protocol. The verifier's public-key consists of a commitment to a pseudorandom function. In the first stage of the protocol, the verifier provides a zero-knowledge argument of knowledge that it knows the decommitment. (Since the prover is resettable and plays the verifier in this subprotocol, the argument used must be resettably-sound.) In the second stage of the protocol, the prover proves that the common input G is Hamiltonian by running the basic proof of Hamiltonicity in parallel. However, the verifier must be prevented from obtaining answers to more than one query for a single series of prover commitments. (Otherwise, a cheating verifier can extract a cycle by resetting P.) This is achieved by having the verifier choose its queries by applying the pseudorandom function, committed to in the public-key, to the prover's commitments. In order to prevent the verifier from cheating, it continues by proving that it indeed computed the query correctly, where like before, the proof used is a resettably-sound zero-knowledge argument.[27] Notice that in essence, the verifier's query string is determined by its public-key and the prover's set of commitments. Intuitively, this prevents a cheating verifier from gaining anything in a resetting attack. We note that given the pseudorandom function, the simulator can generate commitments that it can answer (without accessing the verifier V^*). Thus, the key point in the simulation is showing that the pseudorandom function can be extracted. Having discussed the simulation strategy, we now briefly mention an important point regarding soundness. The verifier V's instructions in our protocol ensure that V never applies the

[26] That is, for every $x \in L$, the verifier V outputs ACCEPT after interacting with P.

[27] If we had used a Verifiable Pseudorandom Function [28] then this second resettably-sound zero-knowledge argument would not have been needed. But since the first resettably-sound zero-knowledge argument of knowledge cannot be avoided (even for VRFs), there is no point in using this stronger primitive here.

pseudorandom function to the same value twice (even if the prover tries to prove the same theorem twice and uses the same commitments in the second stage). Therefore, the verifier's queries as determined by the pseudorandom function are indistinguishable from the case that the verifier chooses its queries at random (and independently of other sessions).

Notation: Let $\{f_s\}_{s\in\{0,1\}^n}$ be a pseudorandom function ensemble.

The public key: Let V be a verifier with identifier id. Then, its public-key (denoted PK_{id}), consists of a perfectly-binding commitment to a random n-bit string (to be used as a seed to the pseudorandom function). That is, $PK_{id} = c$ where $c = C(s; r)$. The associated *secret key* (denoted SK_{id}) consists of the pair (s, r).

Protocol 6.2 (rZK Argument of Knowledge of Hamiltonicity):

- Common Input: *A directed graph* $G = (V_G, E_G)$ *with* $n \stackrel{\text{def}}{=} |V_G|$, *and a public-file* F *consisting of pairs* (id, PK_{id}).
- Auxiliary Input for V (with identity id): *the secret-key* SK_{id}.
- Auxiliary Input for P: *a directed Hamiltonian Cycle,* $C \subset E_G$, *in* G.
- Fixed Randomness for P: $\omega = (\omega_1, \omega_2, \omega_3) \in_R \{0,1\}^{3n}$.
- Stage 0: V Sends a Unique Session Identifier: V *chooses a random string* $R \in_R \{0,1\}^n$ *(to be used as a unique session identifier) and sends it to* P, *along with its identity id.*
- Stage 1: V Proves Knowledge of SK_{id}

  V and P run a resettably-sound zero-knowledge argument of knowledge in which V proves that it knows (s, r) such that $c = C(s; r)$. The randomness used by P when it plays the verifier equals ω_1.

- Stage 2: P Proves that G is Hamiltonian.

  1. *P selects n ("random") permutations $\pi_1, \ldots, \pi_n$ of the vertices V_G and sends the verifier V (perfectly binding) commitments to the adjacency matrices of the resulting permuted graphs. That is, P sends an n-by-n matrix of commitments so that the $(\pi_i(j), \pi_i(k))$'th entry is a commitment to 1 if $(j,k) \in E$ and is a commitment to 0 otherwise. Denote by C_i the matrix of commitments corresponding to the adjacency matrix of $\pi_i(G)$.*

     The randomness for choosing the permutation and computing the commitments is obtained by applying a pseudorandom function, keyed by ω_2, to G and the verifier's public-key PK_{id}.
  2. *For every i $(1 \le i \le n)$, V chooses a query bit by computing $q_i = f_s(R, C_i, i)$ (recall that s is the string committed to in the public file and R is the random session-identifier chosen by V in Stage 0). V sends $q \stackrel{\text{def}}{=} q_1 \cdots q_n$ to P.*[28]

[28] The reason that V also applies the pseudorandom function to R is to ensure that the queries in different sessions are (computationally) independent of each other. We note that if we do not include this session-identifier in the computation of the queries, then the protocol is NOT sequentially-sound. This is because a cheating prover may exploit the fact that he has learned values of $f_s(\cdot)$ in *previous* sessions in order to cheat in the current session. By including R in the computation of q_i, we have that the values of $f_s(\cdot)$ from previous sessions are computationally independent of the its values in the current session. (Recall that in proving soundness we consider an honest verifier. Therefore, with overwhelming probability, the session-identifiers of different sessions are different.)

We also note the reason why the index i is included in the computation of the pseudorandom function. This is to prevent the verifier from replying with the same query bit in the case that the prover sends the same commitment n times. Otherwise, the prover could successfully cheat with probability 1/2.

3. V proves to P that it chose the queries correctly. That is, it proves that

$$\exists(s,r) \text{ s.t. } c = C(s;r) \text{ and for every } i,\ q_i = f_s(R, C_i, i)$$

using a resettably-sound *zero-knowledge proof system. The randomness used by P when it plays the verifier equals ω_3.*

4. If P accepts the proof, then it replies to the queries q_i (for every $1 \le i \le n$) as follows.

- *If $q_i = 0$, then P sends π_i along with the decommitments to the matrix C_i.*
- *If $q_i = 1$, then P decommits to the entries $(\pi_i(j), \pi_i(k))$ of the matrix C_i with $(j,k) \in C$.*

5. For every i, $1 \le i \le n$, V checks P's replies as follows.

- *If $q_i = 0$, then V checks that the revealed graph is isomorphic to G (with isomorphism π_i).*
- *If $q_i = 1$, then V checks that all revealed values are 1 and that the corresponding entries form a simple n-cycle.*

In both cases V also checks that the decommitments supplied by P are proper.

V accepts if and only if all the above conditions hold.

Theorem 6.3 *Protocol 6.2 is a sequentially-sound system of* rZK *arguments in the public-key model.*

Proof: It is easy to see that the protocol is complete (i.e., if the graph is Hamiltonian and both P and V are honest, then V always accepts). We continue by showing that the protocol is sequentially sound (with respect to arbitrary polynomial-size provers).

Sequential Soundness: The differences between the above protocol and the (parallelized) basic proof of Hamiltonicity is that V uses a pseudorandom function to choose its queries, and provides two zero-knowledge proofs related to this pseudorandom function. Intuitively, since the function used by V to choose its queries is pseudorandom and V never applies the function to the same value twice, P^* cannot distinguish these choices from the behavior of a verifier in the basic proof of Hamiltonicity (who chooses its queries independently and at random). (The fact that V never applies the function to the same value twice is guaranteed by having V include the random session-identifier and the index i of the commitment in the computation of the query.) Furthermore, the zero-knowledge proofs are simulatable and thus cannot help P^* learn anything about the pseudorandom function. Therefore the soundness of the proof system can be reduced to the soundness of the parallelized basic proof of Hamiltonicity (for which soundness is known to hold). We now proceed with the formal proof.

Assume by contradiction that there exists a (deterministic) polynomial-size P^* executing a sequential-attack on V, such that with non-negligible probability, in one of the rounds the common-input is a graph G that is not Hamiltonian and yet V outputs ACCEPT. Then, we show how to use P^* to contradict the sequential soundness of the basic parallel proof of Hamiltonicity in the standard model (i.e., where there is no public-key). Specifically, we show how to construct a cheating prover P^{**}, who executes an analogous sequential attack in the standard model and with non-negligible probability succeeds in convincing a verifier V_{basic} that G is Hamiltonian, where V_{basic} is the verifier specified by the parallelized version of the basic proof of Hamiltonicity. (We note that sequential soundness in the standard model follows immediately from "one-time" soundness.)

We now define a prover P^{**} for the parallel version of the basic proof of Hamiltonicity. P^{**} internally incorporates P^* and uses P^* to cheat in its interaction with V_{basic}. This is done by having P^{**} create an interface between P^* (who works according to Protocol 6.2) and V_{basic} (who works according the parallel basic proof of Hamiltonicity), such that each side sees only the messages it expects to see. Thus, P^{**} emulates the verifier messages of Protocol 6.2 that do not belong to the basic proof of Hamiltonicity (this actually includes all messages except for the string of queries $q = q_1 \cdots q_n$). Furthermore, P^{**} only forwards to V_{basic} messages belonging to the basic proof of Hamiltonicity.

We note that P^{**} internally incorporates P^* and thus has both "internal", emulated communication with P^* and real (external) communication with a verifier V_{basic}. The prover P^* is a function of the public-file, the input graph and the series of incoming messages received by P^* during a protocol execution. Thus, formally, P^{**} obtains P^*'s next message by computing $P^*(F, h, G, \overline{m})$ where F is the public-file, h some history transcript (representing messages from previous sessions of the sequential attack), G the current statement being proven and $\overline{m}$ a series of incoming messages to P^*.[29] P^{**} works as follows:

- *Initialization:* P^{**} chooses a random seed s and computes $pk = \text{Commit}(s)$. The commitment pk is then used by P^{**} as a public-key PK_{id} for the simulated interface with P^*. That is, P^{**} defines a public file $F = \{(id, pk)\}$.[30]
- Let h_i denote the history of messages received by P^* until the i'th round – in the first iteration h_1 is empty. Then, for every round of the sequential attack executed by P^{**}:
  1. P^{**} obtains the statement G_i to be proven by P^* in this round and forwards it to V_{basic}. Formally, P^{**} obtains G_i by computing $P^*(F, h_i)$.
  2. Emulating Stage 0, P^{**} sends a random string $R_i \in_R \{0,1\}^n$ to P^*.
  3. P^{**} (internally) plays V's role in Stage 1 and proves the resettably-sound zero-knowledge argument of knowledge of the decommitment of pk, with $P^*(F, h_i, G_i, R_i)$ as the verifier. P^{**} can do this because it indeed knows the decommitment for pk in F. Denote the resulting transcript by t_{pok}. (We stress that this proof is not simulated but is properly executed by P^{**}.)
  4. Entering the emulation of Stage 2, P^{**} (internally) obtains a message from P^* and forwards it to V_{basic} (this message "should" consist of commitments to n adjacency matrices).
  5. P^{**} then receives an n-bit string from V_{basic}, denoted q.
  6. P^{**} runs the zero-knowledge simulator for the zero-knowledge proof of Step 3 in Stage 2 where the verifier is defined by $P^*(F, h_i, G_i, R_i, t_{pok}, q)$. (I.e., here P^* simulates V's role in Step 3 of Stage 2.) Denote the resulting transcript by t_{pf}.
  7. Finally, P^{**} (internally) obtains a message from P^* and forwards it to V_{basic} (this message should consist of answers to the query string q).

     Set $h_{i+1} = (h_i, G_i, R_i, t_{pok}, q, t_{pf})$.

We note the following differences between P^*'s view in a real execution with V and its view in the interface provide by P^{**} in its interaction with V_{basic} (from here on, we refer to this interface by "the emulation by P^{**}"):

[29] We note that some of this is redundant as by the definition of a sequential attack, P^* chooses G on the basis of F and h. However, we include it explicitly for the sake of clarity.

[30] For simplicity, we assume that the public file contains a single identity. This is without loss of generality since the public-keys are independent of each other. Therefore, if there exists a cheating prover who succeeds when there are many public-keys, then there also exists a cheating prover who succeeds when there is just one.

- In the emulation by P^{**}, the query string in every round is uniformly chosen (by V_{basic}), rather than being computed by $q_i = f_s(R_i, C_i, i)$.
- In the emulation by P^{**}, the zero-knowledge proof in Stage 2 is simulated rather than real.

Intuitively, despite the above two differences, P^*'s view in a real execution with V is computationally indistinguishable from its view in the simulation by P^{**}. These views are indistinguishable due to the pseudorandomness of the function used to compute verifier queries and due to the indistinguishability of transcripts of real and simulated proofs. Since P^*'s view implicitly includes the verifier's accept or reject decision, and the decision to accept depends solely on the messages from the basic proof of Hamiltonicity, we have that V_{basic} accepts in its execution with P^{**} with negligibly close probability to V in its execution with P^*. By the sequential soundness of the parallel proof of Hamiltonicity, we obtain a contradiction. We now formally show that P^*'s view is indeed indistinguishable in the above two scenarios.

First consider a hybrid, mental experiment, where the query string is chosen by computing $q_i = f_s(R_i, C_i, i)$ (as in the real execution), yet the proof of Stage 2 is simulated (as in the emulation by P^{**}). P^*'s view in the mental experiment is indistinguishable from the real execution by the indistinguishability of simulated proofs for auxiliary input zero-knowledge. (Actually, we replace here polynomial many proofs by simulated ones. Thus, formally we need to consider a hybrid argument for this. This argument is, however, standard and is therefore omitted here.)

On the other hand, we show that the mental experiment (in which the query string is chosen according to the protocol definition) is indistinguishable from the emulation by P^{**} (in which the queries are chosen uniformly at random). There are two key points to take note of here. First, the function $f_s(\cdot)$ is pseudorandom even to a distinguisher given $pk = \text{Commit}(s)$ and a zero-knowledge proof of knowledge of the decommitment of pk.[31] Second, notice that except with negligible probability, the pseudorandom function is always applied to *distinct* values. This is due to the random session identifier R_i that is sent in the beginning of every round and the fact that even if the prover sends the same commitment more than once in the same round, each commitment is associated with a unique index that is included in the computation of $f_s(\cdot)$. Therefore, we have that, except with negligible probability, the honest verifier never computes $f_s(\cdot)$ twice at the same point. Thus, P^* cannot distinguish between the case that $f_s(\cdot)$ is used and the case that independent queries are chosen each time (which is equivalent to the case that a random function is used since the function is always applied to distinct values). In other words, P^* cannot distinguish between the case that it interacts in the mental experiment and the case that it interacts with P^{**} and V_{basic}.

By combining the above we have that P^*'s view in a real execution with V is indistinguishable from its view in the interface provided by P^{**} in the interaction with V_{basic}. Thus the probability that P^{**} succeeds in convincing V_{basic} of a false statement in the basic proof of Hamiltonicity is negligibly close to the probability that P^* succeeds in convincing V of a false statement in Protocol 6.2. This completes the proof of soundness.

Resettable Zero-Knowledge: First, consider a stand alone execution of the above protocol (rather than an execution in a resettable setting). A simulation would proceed as follows. The simulator, given access to the verifier (black box manner would suffice here), extracts in Stage 1

[31] This is shown by first simulating the zero-knowledge argument of knowledge. Then it is shown that a distinguisher for $f_s(\cdot)$ can be used to distinguish a commitment to s from a commitment to s' (where s and s' are two independently chosen random seeds). This is because if $pk = \text{Commit}(s')$ then $f_s(\cdot)$ is pseudorandom (as the commitment is independent of $f_s(\cdot)$). Thus, if a distinguisher could distinguish $f_s(\cdot)$ from a random function when $pk = \text{Commit}(s)$, then we would obtain an algorithm for distinguishing commitments to s from commitments to s'.

SK_{id}. This is possible because Stage 1 is a argument of knowledge of SK_{id} (from verifier to prover). This gives the simulator the seed s (contained in SK_{id}) for the pseudorandom function f_s to subsequently be used by the verifier to specify his queries in this session. Once extraction is completed, S proceeds to generate a sequence of commitments that it can open as required by the verifier (assuming that the verifier's queries are as prescribed in the protocol; i.e., are obtained by applying the pseudorandom function to the corresponding commitments). The simulator S initializes empty lists Q and C, and proceeds in n steps: For $i = 1$ to n, the simulator chooses at random $q_i \in \{0, 1\}$. If $q_i = 0$, then S chooses at random a permutation of the vertices π_i, and prepares a prefectly binding commitment of an adjecency matrix of the graph permuted according to π. Otherwise (i.e., $q_i = 1$), S chooses at random cycle of n vertices and prepares a a perfectly binding commitment of an n by n matrix in which all entries are 0 except for those entries on the random cycle chosen. Let us denote the commitment prepared by c_i. After this preparation, S computes $q'_i = f_s(R, c_i, i)$. If $q'_i = q_i$, then S augments the lists of commitments it can handle (i.e., sets $C = C, c_i$ and $Q = Q, q_i$), and goes on to iteration $i + 1$, else S goes back and chooses another q_i. (The expected number of times each iteration is repeated is ≈ 2.) Next, S sends C to V, and expects to recieve back a sequence of queries $Q' \in \{0, 1\}^n$ (which is expected to equal Q). At this point, V is expected to prove in resettably-sound (zero-knowledge) fashion that indeed Q' was computed correctly by applying the pseudorandom function commited to in the public file which corresponds to SK_{id}. The simulator S, plays here the role of P in the real protocol in the same way that P would (taking no advantage of his knowledge of s or of Q). Finally, if S accepts the proof of V (as P would have), then S needs to reply to the queries about C specified by Q'. However, if $Q = Q'$ (which happends with high probability, otherwise the proof above would fail with high probability) then S is able to supply the answers V is expecting regarding C.

Now, what happends in a resettable setting? Note that the main change in the above argument needed pertains to the simulator's extraction of SK_{id} from the POK[32] of Stage 1 (another point to note is that the fact that the "correct behavior" proof given by the verifier in Stage 2 is resettably-sound and thus remains valid also in case the verifier can reset P). Once SK_{id} is extracted successfuly, the rest of the simulation will follow exactly as above.

The way we extract SK_{id} from the POK of Stage 1 executed in a resettable setting generalizes the way an analogous task is performed in the public-key model protocol of [7]. All that is needed is to generalize the description in [7], which refers to a specific POK (of Stage 1) to an arbitrary (resettably-sound POK).

We now focus on the extraction of SK_{id} in Stage 1, in a resettable setting. What complicates matter is that it is not a single key that need be extracted but many secret keys as for each verifier there are many possible public-keys (or *identity*'s) $id_1, id_2, ..., id_i, ...$ (and their corresponding secret keys $SK_{id_1}, ..., SK_{id_i}, ...$) from which the verifier picks one at Stage 0 of a session. Once a particular key SK_{id} is extracted, all sessions using this key can be simulated with no problem.

Let $LIST$ (initialized to be empty) contain all the id's for which the simulator already extracted SK_{id}. Define $p_{id_i}^{LIST}$ to be the probability (taken over the coin tosses of P and V) that, among all the id's not in $LIST$, the identity to be used in the first session for which V *gets through a complete run of Stage 1*, is id_i. Note that for any $LIST$, the sum of $p_{id_i}^{LIST}$'s is at most 1, but may be much smaller (if the probability that no session with identity out of $LIST$ gets through a complete run of Stage 1 is large). If for a fixed value of $LIST$, for all i such that id_i not in $LIST$, the probability $p_{id_i}^{LIST}$ is negligible, then it is easy for S to simulate an entire run of the resettable protocol due to the following observations.

[32] POK is a common shorthand for *proof of knowledge*, and we slightly abuse it here by referring to Stage 1 that is an *argument* of knowledge.

- All the actions of P during the Stage 1 POK are independent of P's additional knowledge of the witness of $x \in L$ (which P knows but S doesn't) and can be preformed by S exactly as P would do them. Thus, as long as V does not run Stage 1 to completion (say he aborts) and goes on to Stage 2, a session can be simulated profectly by S in the role of P. Thus, as long as for all i such that id_i not in $LIST$, the probability $p_{id_i}^{LIST}$ is negligible, we never get passed Stage 1 for an SK_{id} that S does not know, and such aborted sessions can be simulated perfectly.

- If we do get passed Stage 1 we are (w.h.p) in a session for which $id \in LIST$. Thus the simulator S already knows SK_{id} and can continue to answer as P would have through Stage 2 of this session (as outlined above in the case of standard zero-knowledge).

Thus, the interesting case is when for some i such that id_i is not in $LIST$, the probability $p_{id_i}^{LIST}$ is non-negligible. For simplicity, think that this probability is noticeable (although standard techniques can be applied to remove the dependency of the description on such a false dicotomy [21]).

Suppose we are in this case for some intermediate value of $LIST$. The simulator task is either to extract a new SK_{id} for which id not in $LIST$, or produce a view H of an execution of the entire resettable protocol (either way progress is made). The (partial) view H is initialized to the empty string. The simulator S starts running acting as the prover would in Stage 1 of any session that gets invoked by the verifier. Whenever the protocol calls for the prover to use his pseudorandom function, the simulator tosses coins, making sure of course that on the same session prefix the same coin are used as would be the case for a true random function. There are a few cases to consider.

- If a session for any id halts midway Stage 1, the simulator extends H to include an aborted transcript of this session.

- If a session for which $id \in LIST$ gets passed Stage 1, then the simulator uses his knowledge of SK_{id} to answer the verifier and gets through Stage 2 of this session. The (partial) view H is extended to contain the transcript of this session.

- If a session for which id is not in $LIST$ gets passed Stage 1, then the simulator tries to obtain the corresponding SK_{id} by using the knowledge-extractor (guaranteed for the POK used in Stage 1). What complicates matters is that the extarctor may get to situations in which it cannot operate (becuase during extraction is may need to proceed in a Stage 2 of a session for which the identity is not yet in $LIST$). The solution is to emulate for the extractor, a knowledge-prover that aborts its execution just after completing Stage 1 of the first session with identity not in $LIST$. Note that this imaginary (or residual) knowledge-prover still succeeds to convince the knowledge-verifier to accept Stage 1 of the session with identity id with probability p_{id}^{LIST}, and thus the extractor will succeed in extracting the corresponding SK_{id} within expected running-time inversly proportional to p_{id}^{LIST}. (Standard tricks, first appearing in [21], have to be applied here; details are omitted.)

  Once the extraction of SK_{id} is completed, the simulator may proceed as in the previous case. Alternatively, it may sets H back to empty and start the entire process again (of course) with the current $LIST$ (which was just extended by one id). In any case, the partial view H is not effected by the extraction process itself.

  We stress that the extraction procedure is invoked only for a fixed polynomial number of times (corresponding to the size of the public file).

- Suppose that all sessions encountered are either using an $id \in LIST$, or never get passed Stage 1, then when V terminates, we finally output H as te simulators view. Indeed, eventually either $LIST$ will contain all the identities of V or we shall arrive to the point that $p_{id_i}^{LIST}$ is negligible for all id_i not in LIST. In either case, we can get through an entire simulation of the resettable protocol w.h.p. and output H.

It remains to formally analyze the running time of this simulator and show that indeed it works in expected polynomial-time. We delay this to the final paper, but remark that the analysis proceeds against the same outline as does the analysis (of the simulator for the public-key rZK protocol) of [7]. ■

6.3 Knowledge Extraction

As we have claimed above, Protocol 6.2 is actually a system of rZK arguments of knowledge. In this section, we briefly describe the extraction procedure, which turns out to be very similar to the extraction procedure for Protocol 5.1. That is, in Protocol 5.1 the verifier is committed to its queries before the prover sends the commitments. Then, in order to extract, the extractor cheats and decommits in more than one way. Likewise here, given the public-key, session-identifier and prover commitments, the verifier is essentially committed to its queries. However, as in Protocol 5.1, the extractor can cheat and send different queries, where the cheating is made possible by running the zero-knowledge simulator in Step 3 of Stage 2, rather than providing a real proof.

Formally, a proper definition of "sequential-extraction in the public-key model" should be provided. (This is analogous to the issue of sequential-soundness which is not implied by one-time soundness in the public-key model.) However, we describe the extractor here in terms of a single execution (recall that above we have shown that *sequential* soundness does hold for Protocol 6.2). We note that one issue that must be resolved is whether or not the extractor has access to a secret-key corresponding to a public-key in the public file. Since the extractor demonstrates that the *prover* knows the witness and the prover does not know the verifier's corresponding secret-key when proving, we have chosen NOT to provide the extractor with this information. Furthermore, this also provides a stronger definition that implies the alternative.

The Extraction Procedure: Recall that it is enough to obtain correct replies to two different queries (for the same set of prover commitments) in order to successfully extract a Hamiltonian cycle. We denote by $P^*(F, G, r, \overline{m})$ the next message sent by P^* where F is the public-file, G the common input graph, r its random-tape and $\overline{m}$ the series of incoming messages to P^*. The extractor K (upon input an encoding of P^*, a public file $F = \{(id, PK_{id})\}$ and a graph G) works as follows:

1. K chooses a uniform string r for the random tape of P^*.

2. Emulating Stage 0, K gives a random string $R \in_R \{0,1\}^n$ to P^*.

3. Emulating Stage 1, K simulates the resettably-sound zero-knowledge argument of knowledge of the decommitment for PK_{id}, where the verifier is defined by $P^*(F, G, r, R)$. Denote the transcript of the proof by t_{pok}.

4. Entering the emulation of Stage 2, K obtains the next message from P^*, which is denoted by $c_1, \ldots, c_n$.

5. K uniformly chooses $q \in_R \{0,1\}^n$ and sends q to P^*. Furthermore, K runs the zero-knowledge simulator for the proof of Step 3 of Stage 2 where the verifier is defined by $P^*(F, G, r, R, t_{pok}, q)$. (We note that the ability of K to provide a *simulated* proof rather than a real proof here, is central to the extraction procedure. Furthermore, recall that for languages outside of $\mathcal{BPP}$, the simulator for a resettably-sound zero-knowledge argument cannot be black-box. Thus, the fact that K is given non black-box access to P^* is essential.)

6. K then obtains the next message from P^* (which should be decommitments).

7. K verifies the proof according to V's instructions.

   (a) If the verification fails, then K halts with output $\perp$.

   (b) If the verification succeeds, then K continues by repeating Steps 5–7 until another success occurs, where in each repetition, the string q is chosen independently and uniformly. Denote the string for which the success occurs by q'. (In each of these repetitions, K uses the same randomness for the zero-knowledge simulation; this technical point simplifies the analysis.)

   If $q' \neq q$, then extraction succeeds and K outputs the cycle C. On the other hand, if $q' = q$, then extraction fails and K outputs $\mathsf{failure}$.

Claim 6.3.1 *Let K be the extractor described above. Then, for every probabilistic polynomial-time machine P^*, every public-file F, every polynomial $p(\cdot)$ and all sufficiently large graphs G,*

$$\Pr[K(\mathsf{desc}(P^*), F, G) \in \mathrm{HAM}(G)] > \Pr[\langle P^*, V\rangle(F, G) = \text{ACCEPT}] - \frac{1}{p(|G|)}$$

where $\mathrm{HAM}(G)$ *denotes the set of Hamiltonian cycles in G.*

Proof Sketch: We first claim that the probability that K halts with output $\perp$ is negligibly close to the probability that V rejects a proof upon interaction with P^*. Notice the following differences between a transcript resulting from a real interaction between P^* and V and a transcript resulting from Steps 1–6 of K's program above:

- The zero-knowledge proofs are simulated and not real.
- The query string is independent of the public-key and its corresponding pseudorandom function.

In the proof of soundness of Protocol 6.2 above, we have shown almost the same claim (the only difference being that there the first zero-knowledge proof was real and not simulated – this difference is not, however, significant). Thus, indistinguishability holds. Since whether or not V should reject is easily derived from a transcript, we have that the probability that K halts with output $\perp$ is negligibly close to the probability that V rejects a proof upon interaction with P^*.

Next, we claim that K outputs $\mathsf{failure}$ with probability 2^{-n} exactly. This follows using an identical argument as in the proof of Lemma 5.1.1. The same argument follows since Steps 5–7 of the extraction procedure here are essentially the same as Steps 3–5 of the extractor described in Lemma 5.1.1.

Now, notice that if K does not output $\perp$ and does not output $\mathsf{failure}$, then it outputs a Hamiltonian cycle. Thus, combining the above together, we have that the probability that K outputs a Hamiltonian cycle is negligibly close to the probability that V accepts in an interaction with P^*.

It remains to show that K is expected polynomial-time. Once again, the same analysis as in Lemma 5.1.1 is correct here and we therefore refer the reader there. ■

Acknowledgements

We are grateful to Ran Canetti for collaboration in early stages of this research. Specifically, Protocol 6.2 is related to some of the ideas that were developed with Ran, with the crucial exception of not anticipating at that time the existence of (constant-round) resettably-sound zero-knowledge arguments of knowledge for $\mathcal{NP}$.

References

[1] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *FOCS 2001*.

[2] B. Barak and O. Goldreich. CS Proofs Under a Standard Assumption. In preparation, 2001.

[3] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Proc. of Crypto'92*, Springer-Verlag LNCS Vol. 740, pages 390–420.

[4] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, Vol. 37, No. 2, pages 156–189, 1988.

[5] M. Bellare and M. Yung. Certifying Permutations: Non-Interactive Zero-Knowledge Based on Any Trapdoor Permutation. *Jour. of Crypto.*, Vol. 9 (3), pages 149–166, 1996.

[6] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS 2001*. Full version, entitled "A Unified Framework for Analyzing Security of Protocols", is available at *Cryptology ePrint Archive*, Report 2000/067.

[7] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. In *32nd STOC*, pages 235–244, 2000. Full version available from *ECCC*, TR99-042, 1999. See also *Cryptology ePrint*, Record 99-22, revised June 2000.

[8] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(\log n)$ Rounds. In *33rd STOC*, 2001.

[9] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *Eurocrypt'00*, pages 418–430, 2000.

[10] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. newblock *SIAM Journal on Computing*, January 2000. A peliminary version appeared in *23rd STOC*, pages 542–552, 1991.

[11] C. Dwork and M. Naor. Zaps and their applications. In *41st FOCS*, pages 283–293, 2000.

[12] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. In *30th STOC*, pages 409–418, 1998.

[13] U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs.* Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990.

[14] U. Feige, A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. *Jour. of Crypto.*, Vol. 1, 1988, pages 77–94.

[15] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SICOMP*, Vol. 29 (1), pages 1–28, 1999.

[16] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

[17] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. In *CRYPTO86*, Springer-Verlag LNCS 263, pages 186–189, 1987.

[18] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.

[19] O. Goldreich. *Foundations of Cryptography: Volume 1 – Basic Tools.* Cambridge University Press, 2001.

[20] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *JACM*, Vol. 33, No. 4, pages 792–807, 1986.

[21] O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, Vol. 9, pages 167–189, 1996.

[22] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM J. Computing*, Vol. 25, No. 1, pages 169–192, 1996.

[23] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38, No. 1, pages 691–729, 1991.

[24] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SICOMP*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[25] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Jour. of Crypto.*, Vol. 7, No. 1, pages 1–32, 1994.

[26] J. Kilian and E. Petrank. Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds In *33rd STOC*, 2001.

[27] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Crypto 2001.*

[28] S. Micali, M.O. Rabin and S. Vadhan. Verifiable Pseudorandom Functions. In *40th FOCS*, pages 120–130, 1999.

[29] S. Micali and L. Reyzin. Min-round resettable zero-knowledge in the public-key model. In *Eurocrypt 2001.*

[30] S. Micali and L. Reyzin. Soundness in the Public-Key Model. In *Crypto 2001.*

[31] L. Reyzin. *Zero-Knowledge with Public Keys.* Ph.D. Thesis, EECS Dept., MIT, June 2001.

[32] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer-Verlag (LNCS 1592), pages 415–431.

Appendix: Blum's Protocol

In the main text, we consider n parallel repetitions of the following basic proof system for the *Hamiltonian Cycle* (HC) problem which is NP-complete (and thus get proof systems for any language in $\mathcal{NP}$). We consider directed graphs (and the existence of directed Hamiltonian cycles).

Construction A.1 (Basic proof system for HC):

- Common Input: *a directed graph* $G = (V, E)$ *with* $n \stackrel{\text{def}}{=} |V|$.
- Auxiliary Input to Prover: *a directed Hamiltonian Cycle,* $C \subset E$*, in* G*.*
- Prover's first step (P1): *The prover selects a random permutation,* π*, of the vertices* V*, and commits (using a perfectly-binding commitment scheme) to the entries of the adjacency matrix of the resulting permuted graph. That is, it sends an* n*-by-*n *matrix of commitments so that the* $(\pi(i), \pi(j))^{\text{th}}$ *entry is a commitment to 1 if* $(i, j) \in E$*, and is a commitment to 0 otherwise.*
- Verifier's first step (V1): *The verifier uniformly selects* $\sigma \in \{0, 1\}$ *and sends it to the prover.*
- Prover's second step (P2): *If* $\sigma = 0$ *then the prover sends* π *to the verifier along with the revealing (i.e., preimages) of all commitments. Otherwise, the prover reveals to the verifier only the commitments to entries* $(\pi(i), \pi(j))$ *with* $(i, j) \in C$*. In both cases the prover also supplies the corresponding decommitments.*
- Verifier's second step (V2): *If* $\sigma = 0$ *then the verifier checks that the revealed graph is indeed isomorphic, via* π*, to* G*. Otherwise, the verifier just checks that all revealed values are 1 and that the corresponding entries form a simple* n*-cycle. In both cases the verifier checks that the decommitments are proper* (*i.e., that they fits the corresponding commitments*)*. The verifier accepts if and only if the corresponding condition holds.*

Proposition A.2 *The protocol which results by* n *parallel repetitions of Construction A.1 is a proof of knowledge of Hamiltonicity with knowledge error* 2^{-n}*.*