Isolated Proofs of Knowledge and Isolated Zero Knowledge

Ivan Damgård[1], Jesper Buus Nielsen[1] and Daniel Wichs[2]

[1] University of Aarhus, Denmark
[2] New York University, USA

Abstract. We introduce a new notion called ℓ-isolated proofs of knowledge (ℓ-IPoK). These are proofs of knowledge where a cheating prover is allowed to exchange up to ℓ bits of communication with some external adversarial environment during the run of the proof.

Without any additional setup assumptions, no witness hiding protocol can be an ℓ-IPoK for *unbounded* values of ℓ. However, for any *pre-defined* threshold ℓ, and any relation in NP and we construct an ℓ-IPoK protocol for that relation. The resulting protocols are zero knowledge (ZK) in the standard sense, i.e., w.r.t. a verifier that communicates only with the prover during the proof. The cost of having a large threshold ℓ is a large communication complexity of the constructed protocol. We analyze these costs and present a solution that is asymptotically optimal.

If a cheating verifier is allowed to communicate arbitrarily with an external environment, it is not possible to construct an ℓ-IPoK that is also ZK with respect to such a verifier. As another new notion, we define ℓ-isolated zero knowledge (ℓ-IZK) where the verifier is ℓ-isolated. For every relation in NP and every ℓ, we construct an ℓ-IPoK protocol that is also ℓ-IZK.

We describe several applications of ℓ-IPoK protocols under the physical assumption that one can ℓ-isolate a prover for the duration of the proof phase. Firstly, we can use a witness indistinguishable (WI) ℓ-IPoK to prevent "man-in-the-middle" attacks on identification schemes. Prior results for this scenario required all verifiers to register keys under a PKI, or the ability to fully isolate the prover. Secondly, a partially isolated prover can register a public key and use a WI ℓ-IPoK to prove knowledge of the corresponding secret key to another party acting as a verifier. This allows us to set up a PKI where the key registrant does not need to trust the Certificate Authority. The PKI is not perfect since the proof is only witness indistinguishable and not zero knowledge. In a companion paper, we show how to set up such a PKI and use it to implement arbitrary multiparty computation securely in the UC framework without relying on any trusted third parties.

Table of Contents

Isolated Proofs of Knowledge and Isolated Zero Knowledge 1
Ivan Damgård, Jesper Buus Nielsen and Daniel Wichs
1 Introduction 3
2 Σ-Protocols 5
3 Isolated Proof of Knowledge and Isolated Zero-Knowledge 6
3.1 Relationship Between the Two Notions of Soundness 9
4 Some Impossibility Results 11
4.1 Impossibility of ∞-IPoK and ∞-IZK 11
4.2 Σ-Protocols are (only) expected ℓ-IPoK, (only) for Very Small ℓ 11
4.3 Impossibility of Constant-Round Black-Box IPoK 12
5 Some Possibility Results 13
5.1 A Simple Black-Box ZK IPoK for NP 13
5.2 A Constant Overhead, Black-Box ZK IPoK for NP 14
Communication Complexity 16
Completeness 16
Extractor 16
The ZK Simulator 18
5.3 Constant-Round, Non-Blackbox IPoK and IZK for NP 18
Constant-Round, Non-Blackbox IZK for NP 18
From IZK to IPoK + IZK 20
Resettable-IZK IPoK 21
5.4 Constant-Overhead, Constant-Round IPoK for NP using a Random Oracles 22
5.5 From IPoK + WI to IPoK + IZK 23
6 Applications of WI IPoK 24
6.1 Preventing "Man-in-the-Middle" Attacks on Identification Schemes 24
6.2 Setting Up a PKI for General UC MPC 24
7 Future Directions 25
A A Computational Σ-Protocol for any PPT Relation 27
B Σ-Protocols with Large Challenge Space are ℓ-IPoK for Very Small ℓ 27
C Proof of Theorem 3 31
D Technical Lemmas for the Proof of Theorem 5 31

1 Introduction

A proof of knowledge [GMR85,BG92], is a protocol where a prover demonstrates to a verifier that he has a certain piece of information - typically the witness for some instance of an NP relation. Soundness of such a proof is usually formalized by insisting that there is a way to extract the witness from any prover who successfully convinces the verifier. The definition implicitly assumes that the prover talks to no one else during the proof. Intuitively, this may seem necessary to ensure that it is the prover himself who knows the witness, and not someone else who helps the prover convince the verifier.

Nevertheless, in this paper we will consider a cheating prover who, during the proof, is able to communicate with some external adversarial entity, called the environment. We will insist that knowledge soundness still means that a witness can be extracted *from the prover himself*. From a technical point of view this means that an extractor is allowed to rewind the prover, but not the environment.

When the cheating prover can communicate *arbitrarily* with the environment, this notion can only be achieved by trivial protocols where the prover essentially hands the witness to the verifier. The obvious reason is that the witness may be located in the environment and the cheating prover only acts as a channel between the environment and the verifier while the environment gives an honest proof. In such a case, the cheating prover learns nothing more than the honest verifier during the proof and hence extraction implies that the honest verifier always learns a witness from a single run of the protocol. This simple attack requires the prover and the environment to communicate an entire transcript of an honest proof. We study what happens when such an attack is prevented by limiting the communication between the prover and the environment to be shorter than the communication used in the protocol.

One can imagine many ways such a partial isolation could be achieved in practical scenarios. If the prover is in close proximity to the verifier, they can be expected to communicate orders of magnitude faster than the prover can communicate with its environment. If in very close proximity, the fixed speed of light alone can be used to isolate a prover. Alternatively, consider a prover implemented on a smart card: for example, a smart card performing an identification protocol. The card reader could try to shield the card completely (i.e. using a Faraday cage) but this requires significant resources. It might be much easier to only prevent *large* amounts of communication. For example, the card reader could limit the amount of communication by measuring the energy consumption of the card. A significant amount of communication takes up a noticeable amount of energy, typically orders of magnitudes larger than what the card needs for standard operation.

To facilitate a formal study of such settings, we propose a notion of ℓ-isolated proofs of knowledge (ℓ-IPoK), where the cheating prover is restricted to communicating only ℓ bits of information with the environment during the run of the proof. Note that the number of bits of information communicated does not necessarily correspond to physical bits. For example, if the prover and the environment share (very) well synchronized clocks, then a short signal can communicate many bits of information based on the time it is sent. Later, we will also see that some of our protocols only need to restrict the number of exchanged *messages* where each message may contain arbitrarily many bits of information.

The case $\ell = 0$ is essentially a normal proof of knowledge, and is therefore well understood: for every NP relation there exists a zero-knowledge 0-IPoK protocol, under standard cryptographic assumptions. In practice, the physical setting determines the level of isolation and hence the communication threshold ℓ. For any such threshold, we would like to construct an ℓ-IPoK protocol. We therefore consider the notion of an IPoK, which is a parameterized compiler that generates an ℓ-IPoK protocol for any value of ℓ polynomial in the security parameter κ. Letting C denote the communication complexity of the generated proof system, we call $O = C/\ell$ the *overhead*. We saw that any non-trivial ℓ-IPoK protocol must have $C > \ell$ so an overhead greater than 1 is necessary. We will focus on making the overhead as small as possible. We will construct IPoK compilers that are non-trivial in that they are Zero Knowledge (ZK) in the standard sense (when the verifier is fully isolated), or at least witness indistinguishable (WI).

It turns out to be relatively simple to construct an IPoK with an overhead of $O = \text{poly}(\kappa)$ and with $\mathcal{O}(\ell + \kappa)$ rounds of communication. This is done by repeating a standard Σ-protocol $\ell + \kappa$ times so that there are many iterations where the prover cannot consult the environment and therefore can only survive if he knows a witness. We introduce several novel techniques that allow us to construct significantly more efficient IPoK compilers. Under appropriate cryptographic assumptions discussed later, we show the following:

1. All NP relations R have a ZK IPoK proof system with constant overhead (but not constant round).
2. All NP relations R have a ZK IPoK proof system that is constant-round (but not constant overhead).
3. Under a non-black-box assumption on cryptographic hash-functions, all NP relations R have a ZK IPoK proof system that is constant overhead and constant-round. This assumption holds in the non-programmable random oracle model.

Result 1 uses only black-box techniques and is optimal in the sense that we can prove (Theorem 3) that no witness hiding IPoK which is black-box extractable can run in a constant number of rounds. In fact, the number of rounds has to grow with ℓ. Result 2 uses non-black-box techniques, but only makes standard assumptions. Since all known approaches to using non-black-box techniques based on standard assumptions are highly inefficient, there is little hope of using this method to get a compiler with constant overhead. Result 3 then uses the non-programmable random oracle model as a non-black-box technique, which results parameters that are nearly optimal in all respects.

We also propose a notion of ℓ-isolated zero-knowledge (ℓ-IZK), where we require that a simulator can simulate any cheating verifier V^* that communicates at most ℓ bits with its environment during the proof. Since 0-IZK is essentially equivalent to the standard notion of ZK, it is known that every relation in NP has a 0-IPoK, 0-IZK protocol. We will give a simple argument showing that ℓ-IZK proof of knowledge protocols with $\ell \geq C$ only exist for trivial languages. On the positive side, we show how to construct an ℓ-IZK, ℓ-IPoK protocol for any NP relation R and any ℓ polynomial in the security parameter κ.

We conclude the paper by mentioning some applications of ℓ-IPoK using the physical assumption that one can ℓ-isolate a prover for the duration of the proof phase. Firstly, we can use a witness indistinguishable (WI) ℓ-IPoK to prevent "man-in-the-middle" attacks on identification schemes. Previous results for this scenario required either the ability to completely isolate the prover, or a PKI in a very strong sense, where all verifiers must register a public key for which they know the corresponding secret key. Secondly, a partially isolated prover can register a public key and use a WI ℓ-IPoK to prove knowledge of the corresponding secret key to another party acting as a verifier. This allows us to set up a PKI where the key registrant does not need to trust the Certificate

Authority. The PKI is not perfect since the proof is only witness indistinguishable and not zero knowledge. In a companion paper, we show how to set up such a PKI and use it to implement arbitrary multiparty computation securely in the UC framework without relying on any trusted third parties. In some sense, this justifies our choice of considering partially isolated parties for proofs of knowledge only rather than studying arbitrary multiparty computation in general, since the latter follows from the former.

2 Σ-Protocols

An NP relation R is a set of pairs (x, w) where $(x, w) \stackrel{?}{\in} R$ can be checked in poly-time in the length of x. For such a relation we define the witnesses for an instance x as $W_R(x) = \{w | (x, w) \in R\}$ and the language $L(R) = \{x | W_R(x) \neq \emptyset\}$.

We use Σ-protocols throughout the paper. A Σ-protocol is given by four PPT ITMs $(P, V, \mathcal{S}, \mathcal{X})$. In a Σ-protocol for relation R, the prover P is given $(x, w) \in R$ and the verifier V is given x. The protocol has three rounds: the prover $P(x, w)$ sends the first message a, the verifier $V(x)$ sends a uniformly random challenge $e \in \{0, 1\}^l$, and P returns a response z. At the conclusion of the protocol, $V(x)$ outputs a judgment $J = \texttt{accept}$ or $J = \texttt{reject}$ based only on the conversation (x, a, e, z). An accepting conversation (x, a, e, z) is one for which V outputs accept. A Σ-protocol is called complete for R if $P(x, w)$ and $V(x)$ always produce accepting conversations. It is called special knowledge sound for R if, given two accepting conversations (x, a, e, z) and (x, a, e', z') with $e \neq e'$, the extractor $\mathcal{X}$ outputs $w = \mathcal{X}(x, a, e, z, e', z')$ such that $(x, w) \in R$. It is called special honest verifier zero-knowledge for R if for all $(x, w) \in R$ the simulator $\mathcal{S}$ on input (x, e) produces a simulated conversation (x, a, e, z) which is computationally indistinguishable from a conversation produced by $P(x, w)$ on challenge e. It is called statistical special honest verifier zero-knowledge for R if the distribution of simulated conversations is statistically close to the distribution of conversations produced by (P, V). A Σ-protocol is called a (statistical) Σ-protocol for R if it is complete, special knowledge sound and (statistical) special honest verifier zero-knowledge for R. We call it a Σ-protocol with large challenge space if the number of possible challenges is superpolynomial in the security parameter κ, so that $e \in \{0, 1\}^l$ where $l = \omega(log(\kappa))$.

Many relations in cryptography have statistical Σ-protocols, but not all NP relations are known to have statistical Σ-protocols. If however there exists perfectly binding, computationally hiding commitment schemes then all NP relations have a Σ-protocol with computational special honest verifier zero knowledge. For completeness we recall one such construction in Appendix A. Any Σ protocol can be made into a protocol with large challenge space. This is simply done by choosing κ independent first messages $a = (a_1, \ldots, a_\kappa)$, then having κ challenges $e = (e_1, \ldots, e_\kappa)$ and giving κ responses $z = (z_1 \ldots, z_\kappa)$. It is easy to verify that this protocol is complete, special knowledge sound and special honest verifier zero-knowledge for R if the original protocol is as well.

Given two NP relations R_1 and R_2 one can define $R = R_1 \vee R_2$ by $((x_1, x_2), w) \in R$ iff $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$. Given two Σ-protocols Σ_1 and Σ_2 for R_1 respectively R_2 one can use the OR-construction [CDS94] to construct a Σ-protocol $\Sigma = \Sigma_1 \vee \Sigma_2$ for $R_1 \vee R_2$. This Σ-protocol will in addition be witness indistinguishable (WI) in the sense that a proof with instance x using witness w_1 is (at least computationally) indistinguishable from a proof with instance x using witness w_2 for an arbitrary (PPT) cheating verifier V^* – even if V^* is given w_1 and w_2. This in turn implies that the proof is witness hiding (WH) if the relations are hard: A cheating verifier which can compute a witness for R with non-negligible probability p, after seeing a proof, by definition computes a witness for either R_1 or R_2 with probability p. If we let P use a random witness, $w_l \in \{w_1, w_2\}$, then because of WI, the cheating verifier will compute the witness w_{3-l} not used by P with a probability negligibly close to $p/2$. This would contradict the hardness of R_{3-l}.

3 Isolated Proof of Knowledge and Isolated Zero-Knowledge

We start by introducing the notions of ∞-IPoK and ∞-IZK, and then discuss how to restrict the communication. An interactive proof system is defined by the PPT ITMs (P, V). We define the following notions:

Completeness For a proof system (P, V) we define completeness by letting some PPT environment $\mathcal{Z}$ pick $(x, w) \in R$ and then running (P, V) on (x, w). We require that V accepts with all but negligible probability. For simplicity we consider only protocols running in some fixed number of rounds ρ. The honest execution proceeds as in Fig. 1. We require that $\Pr[\textsc{Exec}^R_{P,V,\mathcal{Z}}(\kappa) = 0]$ is negligible in κ for all PPT $\mathcal{Z}$.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then (x, w) is input to P and x is input to V.

execution: Then for $r = 1, \ldots, \rho$ the verifier V is activated to produce a message $v^{(r)}$ that is input to P which is activated to produce a message $p^{(r)}$ that is input to V. Then V is activated to produce a judgment $J \in \{\texttt{accept}, \texttt{reject}\}$. The output of the execution is a bit $\textsc{Exec}$, where $\textsc{Exec} = 1$ iff $J = \texttt{accept}$.

Fig. 1. Execution $\textsc{Exec}^R_{P,V,\mathcal{Z}}(\kappa)$ with honest parties.

Knowledge Soundness To define knowledge soundness, we model a cheating prover by replacing P with an arbitrary PPT ITM P^*. We assume that the cheating prover is able to communicate with its environment during the attack on V. This models the scenario where a cheating prover, for instance, might be able to observe surrounding protocols (of which the proof might be a part) and might be a part of a larger coordinated attack on these protocols. In addition we now allow the environment to pick x which is not necessarily in $L(R)$. We augment the game with a PPT extractor $\mathcal{X}$ whose goal is to recover the witness w. It is unreasonable to assume that the extractor is able to compute a witness from the state of the prover alone, since it is possible that the environment simply gives the witness to the prover as part of their communication and hence the prover does not "know" the witness at the beginning of the protocol. However, even in such a scenario, the prover does know a witness at the conclusion of the proof and hence we should allow it.

Strong knowledge soundness: One option for defining knowledge soundness is to require that the witness $w \in W_R(x)$ can be computed from the view of the prover (including its random coins and its communication with the verifier V and environment $\mathcal{Z}$) at the conclusion of any accepting run of the protocol. The extraction game is outlined in Fig. 2. We define strong knowledge soundness by requiring that for each PPT environment $\mathcal{Z}$ and each PPT cheating prover P^* there exists a PPT extractor $\mathcal{X}$ such that $\Pr[\textsc{Extr}^R_{P^*,V,\mathcal{Z},\mathcal{X}}(\kappa) = 0]$ is negligible in κ.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce x, and x is input to P^* and V.
execution: Then for $r = 1, \ldots, \rho$ the verifier V is activated to produce a message $v^{(r)}$ that is input to P^* which is activated to produce a message $p^{(r)}$ that is input to V. Besides this P^* can at any time output a message y to $\mathcal{Z}$ and get back a reply z. At the conclusion of the ρ rounds, the verifier V produces a judgement $J \in \{\texttt{accept}, \texttt{reject}\}$.
extraction: The output of an execution is a bit $\textsc{Extr}$. If $J = \texttt{reject}$ then $\textsc{Extr} = 1$. Otherwise we construct the view σ to be a concatenation of the random coins of P^*, the messages $v^{(r)}, p^{(r)}$ exchanged between prover and verifier, and all the messages exchanged between prover and environment. We let $w = \mathcal{X}(\kappa, \sigma)$. If $w \in W_R(x)$, then $\textsc{Extr} = 1$ and otherwise $\textsc{Extr} = 0$.

Fig. 2. Strong knowledge soundness extraction: $\textsc{Extr}^R_{P^*,V,\mathcal{Z},\mathcal{X}}(\kappa)$

Simulation based knowledge soundness: As the naming suggests, the above notion of "knowing a witness" is stronger than what is required in many scenarios. We propose an alternative approach that is heavily influenced by the ideas of the Universal Composability framework. We define a real world interaction between a cheating prover, verifier and environment in Fig. 3. We compare this to a simulated execution outlined in Fig. 4 where the extractor $\mathcal{X}$ replaces the machines P^* and V.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce x, and x is input to P^*.
execution: Then for $r = 1, \ldots, \rho$ the verifier V is activated to produce a message $v^{(r)}$ that is input to P^* which is activated to produce a message $p^{(r)}$ that is input to V. Besides this P^* and $\mathcal{Z}$ can communicate arbitrarily. At the conclusion of the ρ rounds, the verifier V produces a judgement $J \in \{\texttt{accept}, \texttt{reject}\}$ which is given to $\mathcal{Z}$. Lastly $\mathcal{Z}$ outputs a bit $\textsc{Exec} \in \{0, 1\}$.

Fig. 3. Real world execution with cheating prover: $\textsc{Exec}^R_{P^*,V,\mathcal{Z}}(\kappa)$.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce x, and x is input to $\mathcal{X}$.
execution: The machines $\mathcal{X}$ and $\mathcal{Z}$ may communicate arbitrarily. At some point, $\mathcal{X}$ produces a judgement $J \in \{\texttt{accept}, \texttt{reject}\}$ and a string $w \in \{0, 1\}^*$. Lastly, J is given to $\mathcal{Z}$ which outputs a bit $\textsc{Sim} \in \{0, 1\}$.

Fig. 4. Simulated execution with extractor: $\textsc{Sim}^R_{\mathcal{X},\mathcal{Z}}(\kappa)$.

The extractor $\mathcal{X}$ is a PPT ITM with protocol access to $\mathcal{Z}$. Specifically, it cannot rewind the environment. However, it can run an internal copy of P^* which it can rewind at will. We have two requirements:

1. The environment cannot distinguish interactions with P^* and V from interactions with $\mathcal{X}$. Formally, $|\Pr[\textsc{Exec}^R_{P^*,V,\mathcal{Z}}(\kappa) = 1] - \Pr[\textsc{Sim}^R_{\mathcal{X},\mathcal{Z}}(\kappa) = 1]|$ is negligible in κ. This ensures that the extractor produces $J = \texttt{accept}$ with roughly the same probability as an honest verifier interacting with the cheating prover, even conditioned on the random coins of the environment.
2. In the simulation game, whenever the extractor gives the judgement $J = \texttt{accept}$, it also output $w \in W_R(x)$ with all but negligible probability.

We define simulation based knowledge soundness by insisting that for any PPT environment $\mathcal{Z}$ and any PPT cheating prover P^*, there exists a PPT extractor $\mathcal{X}$ which satisfies the two requirements above. It is easy to see that an extractor which satisfies the simulation based knowledge soundness definition is necessary and sufficient to UC simulate the interaction between a corrupted prover and an honest verifier.

Most of our positive results will have strong knowledge soundness extractors and we will be explicit when this is not the case. However, to capture the full range of constructions, and to make our impossibility results as strong as possible, we say that a protocol is ∞-IPoK if it has simulation based knowledge soundness. We explore the relationship between the two notions in Section 3.1.

If there exists one $\mathcal{X}$ which works for all provers P^* and all environments $\mathcal{Z}$, and $\mathcal{X}$ only uses rewinding black-box access to P^*, then we say that (P, V) is a black-box ∞-IPoK for R. Sometimes we allow a small cheat and let $\mathcal{X}$ run in expected polynomial time in which case we say that the protocol is an expected ∞-IPoK.

Zero Knowledge We model a cheating verifier by replacing V with an arbitrary PPT ITM V^*. We assume that the cheating verifier is able to communicate with its environment during the attack on P. We model this by allowing V^* to communicate with $\mathcal{Z}$. We assume that the execution stops by $\mathcal{Z}$ outputting a bit. The execution with a cheating verifier is given in Fig. 5.

To define zero-knowledge we compare the execution $\text{Exec}^R_{P,V^*,\mathcal{Z}}$ to a simulation $\text{Sim}^R_{\mathcal{S},\mathcal{Z}}$, where $\mathcal{S}$ is an ITM acting as simulator. We want to capture that the proof does not leak any information on w to V^* which V^* could not have generated itself. We model the information that V^* can collect by what it is able to output to the environment. The job of the simulator $\mathcal{S}$ is then to demonstrate constructively that whatever V^* can leak to the environment could have been computed by V^* *without access to* P. The details are given in Fig. 6. Because simulation using a strict PPT simulator is hard, one usually allows a small cheat by letting $\mathcal{S}$ be expected PPT. We say that (P, V) is ∞-IZK for R if, for every PPT environment $\mathcal{Z}$ and every PPT cheating verifier V^*, there exists an expected PPT simulator $\mathcal{S}$ such that $|\Pr[\text{Sim}^R_{\mathcal{S},\mathcal{Z}} = 1] - \Pr[\text{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa) = 1]|$ is negligible in κ.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then (x, w) is input to P and x is input to V^*.
execution: Then for $r = 1, \ldots, \rho$ the cheating verifier V^* is activated to produce a message $v^{(r)}$ that is input to P which is activated to produce a message $p^{(r)}$ that is input to V^*. Besides this V^* can at any time output a message y to $\mathcal{Z}$ and get back a reply z. The execution stops by $\mathcal{Z}$ outputting a bit $\text{Exec} \in \{0, 1\}$.

Fig. 5. Execution $\text{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa)$ with a cheating verifier.

setup: First all entities are given κ. Then $\mathcal{Z}$ is run to produce $(x, w) \in R$. Then x is input to $\mathcal{S}$.
execution: Then $\mathcal{S}$ can at any time output a message y to $\mathcal{Z}$ and get back a reply $z \in \{0, 1\}^*$. The execution stops by $\mathcal{Z}$ outputting a bit $\text{Sim} \in \{0, 1\}$.

Fig. 6. Simulation $\text{Sim}^R_{\mathcal{S},\mathcal{Z}}(\kappa)$

Isolation The above definition of ∞-IZK, ∞-IPoK is equivalent to that of universally composable zero knowledge proofs of knowledge, as the cheating party is allowed arbitrary communication with its environment. We now describe how a corrupted party is isolated from its environment. We start with the cheating prover in Fig. 2. We do not restrict how much P^* and $\mathcal{Z}$ communicate before or after the proof phase. However, from the point where P^* receives $v^{(1)}$ until it outputs $p^{(\rho)}$ we require that P^* and $\mathcal{Z}$ communicate by writing on two designated tapes IN and OUT. The tape IN is left-to-right write-only for $\mathcal{Z}$ and is read-only for P^* while the tape OUT is left-to-right write-only for P^* and read-only for $\mathcal{Z}$. We say that P^* is $(\ell_{\mathcal{Z}}, \ell_P)$-isolated if P^* never writes beyond position ℓ_P on OUT and never reads beyond position $\ell_{\mathcal{Z}}$ on IN. We say that P^* is ℓ-isolated if it is (ℓ, ℓ)-isolated. We restrict the cheating verifier in Fig. 5 in the same way, counting its communication with $\mathcal{Z}$ from sending $v^{(1)}$ until receiving $p^{(\rho)}$. We then say that (P, V) is an $(\ell_{\mathcal{Z}}, \ell_P)$-IPoK for R if in the definition of knowledge soundness we restrict ourselves to $(\ell_{\mathcal{Z}}, \ell_P)$-isolated cheating provers P^*. Similarly, we say that (P, V) is $(\ell_{\mathcal{Z}}, \ell_P)$-IZK for R if we restrict the definition of zero knowledge to only $(\ell_{\mathcal{Z}}, \ell_P)$-isolated cheating verifiers V^*. We define black-box and expected notions as above. We use ℓ-X to denote (ℓ, ℓ)-X.

Finally, we define the notion of a parameterized protocol which takes the security parameter κ and the isolation parameter ℓ (polynomial in κ) as inputs and produces a protocol which is secure under ℓ-isolation. A parameterized IPoK for R produces an ℓ-IPoK. We often simply say IPoK to refer to a parameterized IPoK protocol. Letting $C(\kappa, \ell)$ denote the communication complexity of the produced ℓ-IPoK, we use $C(\kappa, \ell)/\ell$ to denote the overhead of the IPoK. A parameterized IZK, or just an IZK, for R takes κ, ℓ as inputs, and produces an ℓ-IZK protocol. The overhead is defined similarly. An IPoK + IZK compiler produces a protocol which is ℓ-IPoK and ℓ-IZK.

3.1 Relationship Between the Two Notions of Soundness

We show that, as the naming implies, strong knowledge soundness is strictly stronger than simulation based knowledge soundness. First assume that, for some environment $\mathcal{Z}$ and prover P^*, there is a strong knowledge soundness extractor $\mathcal{X}$. In Fig. 7, we show how to use $\mathcal{X}$ to construct a simulation based knowledge soundness extractor $\mathcal{X}'$.

1. $\mathcal{X}'$ runs an internal copy of P^* and V. It allows them to interact without interference and forwards messages from $\mathcal{Z}$ to P^* and from P^* to $\mathcal{Z}$. It stores the transcripts of these communications as well as the transcript of the protocol with V.
2. At the conclusion of the protocol, the extractor $\mathcal{X}'$ outputs the judgement J that was produced by the internal copy of the verifier V.
3. If $J = 1$ then $\mathcal{X}'$ constructs the view σ as the random coins of P^*, the transcript of the proof with V and the transcript of the communication with $\mathcal{Z}$. It outputs $w = \mathcal{X}(\kappa, \sigma)$.

Fig. 7. Reduction of simulation based knowledge soundness to strong knowledge soundness

It is easy to see that the simulation is indistinguishable from real world execution; in fact $\Pr[\textsc{Exec}^R_{P^*,V,\mathcal{Z}}(\kappa) = 1] = \Pr[\textsc{Sim}^R_{\mathcal{X},\mathcal{Z}}(\kappa) = 1]$ since $\mathcal{X}'$ faithfully runs the real execution internally. In addition when $J = \texttt{accept}$, then $\mathcal{X}'$ outputs $w \in W_R(x)$ with all but negligible probability since $\mathcal{X}$ computes a witness with all but negligible probability.

We now show that strong knowledge soundness is *strictly* stronger than simulation based knowledge soundness under the assumption that there exists a IND-CPA secure dense public key encryption scheme. Let (E, D) be the encryption and decryption algorithms for such a scheme. Also, let

R be an NP relation which is hard on average.[3] Consider the protocol between $P(w, x)$ and $V(x)$ outlined in Fig. 8.

1. The verifier $V(x, w)$ obliviously samples a random public key pk which it sends to $P(x)$.
2. The prover P sends $C = E_{pk}(w)$ to V.
3. The prover runs a Σ-protocol proof (with large challenge space) that C is indeed an encryption of a witness for x. Formally the protocol Σ will be a proof for the NP relation R' which consists of instances (C, x) and witnesses (r, w) such that $C = E_{pk}(w; r)$ and $(x, w) \in R$. The verifier accepts if the run of Σ is accepting.

Fig. 8. Protocol with only simulation based knowledge soundness.

The protocol is not witness hiding since a cheating V^* can chooses the public key pk so that it does know a corresponding secret key. However, the protocol is HVZK since Σ is HVZK, and the encryption scheme is IND-CPA so step 2 can be simulated by sending $C = E_{pk}(0)$.

We show that the protocol is simulation based knowledge sound but *not* strong knowledge sound with respect to a completely unisolated prover. In the introduction we mentioned the simple attack in which the cheating prover simply acts as a channel for an honest proof performed by the environment. In the definition of strong knowledge soundness, the existence of an extractor which works for such a prover and environment implies that one may extract a witness from the transcript of an honest proof alone. Since the protocol is HVZK, this would contradict the hardness of R and hence a strong knowledge soundness extractor does not exist.

However, the simulation based soundness extractor can run a dishonest proof with P^* by choosing a public key for which it does know a secret key sk (and otherwise acting as an honest verifier). The distribution of such an interaction is equivalent to that of an interaction with an honest V and hence cannot be distinguished by the environment from a real interaction. In addition, the extractor can extract $w = D_{sk}(C)$. It wins with all but negligible probability since, if C is not a valid encryption of the witness, the prover and environment together will fail in giving an accepting run of the Σ-protocol with all but negligible probability.

This example highlights the difference between strong knowledge soundness and simulation based knowledge soundness. One can interpret simulation based soundness as meaning that the environment's communication indeed leaks a witness to the prover, but the prover can choose not to learn it.

In the introduction we mentioned the simple attack in which a prover acts as a channel for an honest proof performed by the environment. We argued that the existence of an extractor that works relative to this attack implies that an honest verifier must learn the witness from an accepting proof. The argument we gave holds for strong knowledge soundness but, as we just saw, does not hold for simulation based soundness. However, we can still use the same reasoning to say that an ∞-IPoK with simulation based knowledge soundness cannot be witness hiding. When the cheating prover simply channels an honest proof performed by the environment, the simulation based knowledge extractor cannot learn more by interacting with the environment than what a cheating verifier could learn from a single run of the proof.

[3] For example, for any one way function f the relation (x, w) where w is a random string and $x = f(w)$. One way functions are implied by IND-CPA public key encryption.

4 Some Impossibility Results

In this section we give impossibility results which points forward to some of the later positive results.

4.1 Impossibility of ∞-IPoK and ∞-IZK

In the introduction (and in Section 3.1) we gave a simple argument showing that one cannot construct a witness hiding ∞-IPoK without setup assumptions. This means that, in order to get some meaningful notion of security, we must isolate the prover. We now show that when the verifier is unisolated, one cannot achieve a zero knowledge proof of knowledge even when the prover is completely isolated. The proof is based on the techniques used in [CKL03].

Theorem 1. *For any relation R in NP that has an (expected) ℓ-IZK, (expected) 0-IPoK protocol with communication complexity $C \leq \ell$, there is an (expected) PPT algorithm which gets an instance x in the language $L(R)$ and outputs a witness w such that $(x, w) \in R$.*

Proof. Consider a corrupted verifier V^* which acts as a channel between the honest prover P and environment $\mathcal{Z}$. The environment $\mathcal{Z}$ internally runs the code of the honest verifier to produce the challenge messages. It outputs 1 if the proof is accepting and 0 otherwise. For any protocol, the completeness property ensures that the environment will output 1 with all but negligible probability. The simulator $\mathcal{S}$ simulating the corrupted verifier in the ideal world only gets x but does not get w. In order for the simulator to succeed, it must run an accepting proof with the environment with all but negligible probability. Hence, since the protocol is 0-IPoK, there is an efficient extractor which can extract a witness from the simulator $\mathcal{S}$ with probability $1 - negl(\kappa)$. The extractor and simulator together form a (expected) PPT algorithm which gets input x and can, with all but negligible probability, extract some witness w.

This result shows us that we cannot get full ZK by just isolating the prover. However, we will show that one can still get non-trivial ℓ-IPoK protocols which are zero knowledge in the standard sense or at least witness indistinguishable.

4.2 Σ-Protocols are (only) expected ℓ-IPoK, (only) for Very Small ℓ

We explore for which isolations one can expect a simple Σ-protocol to be an ℓ-IPoK.

Theorem 2. *If there exist perfectly binding, computationally hiding commitments, even a Σ-protocol with large challenge space for relation R need not be a black-box,* strict *0-IPoK for R. However, any such protocol is black-box, expected $(\mathcal{O}(\log(\kappa)), \infty)$-IPoK for R and also black-box, expected $(\infty, \mathcal{O}(\log(\kappa)))$-IPoK for R.*

A Σ-protocol with large challenge space is a black-box, *strict* PoK in the standard sense and hence this result shows that there is some subtle difference between standard PoK and 0-IPoK. In the standard sense of PoK, the usual strict PPT extractor for a Σ protocol runs the prover once honestly then rewinds and tries one random alternate challenge hoping to get two accepting conversations. It has a noticeable probability of success which is amplified by repeating the above process several times with fresh random coins for the prover.

In our setting this cannot be done since even a 0-isolated cheating prover can communicate with the environment prior to the start of the proof and hence the environment can determine the randomness of the prover. The extractor is prevented from running many independent instances of the proof since it cannot rewind the environment. In Appendix B we give a general result, showing that if there exist perfectly binding, computationally hiding commitments, then there also exists a Σ-protocol which is not a *strict, black-box* 0-IPoK.

The usual *expected* PPT rewinding extractor (in the standard sense) for a Σ protocols simply runs the cheating prover on alternate uniformly random challenges until it answers two of them correctly. This extractor only rewinds to the challenge portion of the proof and does not need to start the prover with fresh randomness. Hence it is also satisfies our definition of an expected PPT strong knowledge soundness extractor for a 0-isolated prover, and so a Σ protocol with a super-polynomial challenge space is indeed an *expected* 0-IPoK.

The intuition behind the results that a Σ-protocol with super-polynomial challenge space is black-box, expected $(\mathcal{O}(\log(\kappa)), \infty)$-IPoK and black-box, expected $(\infty, \mathcal{O}(\log(\kappa)))$-IPoK is that a Σ-protocol is a black-box, expected 0-IPoK and that $\mathcal{O}(\log(\kappa))$ bits of communication should not help a PPT machine as they can be guessed. So, a Σ-protocol is $\mathcal{O}(\log(\kappa))$-IPoK. Then one observes that allowing more communication in one direction does not make the channel strong enough to violate IPoK.

Given these intuition, the proofs are fairly straight forward, and can be found in Appendix B. In the next section we show that once ℓ becomes super-logarithmic, we can no longer expect a Σ protocol to be an ℓ-IPoK, at least not with a black-box extractor.

4.3 Impossibility of Constant-Round Black-Box IPoK

We prove that no black-box IPoK compiler can run in a constant number of rounds. In fact, we prove the following stronger result.

Theorem 3. *Any black-box construction of a witness hiding (expected) IPoK compiler, parameterized by the communication threshold ℓ and the security parameter κ, with ρ rounds of communication must satisfy $\ell/\rho = \mathcal{O}(\log(\kappa))$.*

This is result implies that once ℓ is super-logarithmic ($\ell = \omega(\log(\kappa))$), then no protocol with $\mathcal{O}(1)$ rounds can be a witness hiding ℓ-IPoK.

We give a high level sketch of the proof and defer the full proof to Appendix C. For any protocol with sufficiently few rounds of communication, we define a prover P^* and environment $\mathcal{Z}$ which share a (hardcoded) secret key. The prover has a hardcoded witness w which it uses to follow the protocol honestly. However, it also "checks in" with the environment in each round prior to outputting any protocol message. This is done by P^* computing a short (length ℓ/ρ) digest of its view of the protocol thus far, sending it to $\mathcal{Z}$, and having $\mathcal{Z}$ reply with an authentication tag (also length ℓ/ρ) for this digest using the shared secret key. An extractor can rewind P^* and send some modified challenge. However, this will produce a different view. Hence, in order for the extractor to get any additional information via rewinding, it has to be able to efficiently find a collision (two views with the same digest) or guess the authentication tag. This is only possible if $\ell/\rho = \mathcal{O}(\log(\kappa))$.

5 Some Possibility Results

5.1 A Simple Black-Box ZK IPoK for NP

Given any NP relation R, let Σ be a computational Σ-protocol for R. We present a simple construction of an IPoK compiler for R using the protocol Σ. For any ℓ and κ, let Σ^* be the proof system where Σ is run $\rho = \ell + \kappa$ times in sequence with one-bit challenges: For $r = 1, \ldots, \rho$, first P computes the first message a^r for Σ and sends it to V. Then the verifier sends a uniformly random $e^r \in \{0, 1\}$ and P returns the response z^r to V. The verifier V accepts iff (x, a^r, e^r, z^r) is accepting for all $r = 1, \ldots, \rho$.

Theorem 4. *The proof system Σ^* is an ℓ-IPoK for R. In addition, it is ZK in the standard sense of a fully isolated verifier.*

It is well known that there is an expected PPT simulator which simulates many repetitions of a Σ-protocol with 1 bit challenges for any isolated malicious verifier V^*. Hence the above system is 0-IZK. This also implies that it is witness indistinguishable (WI).

To see that Σ^* is ℓ-IPoK, let P^* be any cheating prover. The strong knowledge soundness extractor (recall Fig. 2) gets the transcript of a random accepting execution. Then, for each $r = 1, \ldots, \rho$, it rewinds P^* to the point just before e^r was sent to P^* and sends $e^{r\prime} = 1 - e^r$ instead. If P^* sends anything to $\mathcal{Z}$, then the extractor aborts the work on round r. Otherwise, it runs P^* (and replays any communication that was sent from $\mathcal{Z}$ to P^* in this stage during the actual proof) and gets a response $z^{r\prime}$. If $(x, a^r, e^{r\prime}, z^{r\prime})$ is accepting, then we can use the special knowledge soundness of Σ to compute $w \in W_R(x)$. Otherwise, the extractor proceeds to the next round. If no round yields $w \in W_R(x)$, then it gives up.

Clearly $\mathcal{X}$ is PPT. We want to show that the probability that P^* yields an accepting execution which $\mathcal{X}$ cannot extract is negligible; We call such an execution a winning execution since on such executions $\mathcal{Z}$ and P^* win the extraction game outlined in Fig. 2.

First let us frame the problem more abstractly. The random coins of P^* and $\mathcal{Z}$ together completely determine a strategy of how P^* responds to the challenges posed by V and the communication exchanged for each such message. We model such a strategy as a binary tree T. The edges of the tree represent the two possible challenges the verifier can send at any point in the protocol. The nodes of the tree represent the state of the prover P^* (and the environment $\mathcal{Z}$) at various stages in the protocol. An execution of the protocol between P^* and V corresponds to a random path from the root of the tree to a leaf.

We call a node e-correct if the prover that finds itself in the state represented by that node gives the correct response (one on which the verifier does not reject) for the challenge bit $e \in \{0, 1\}$, possibly after conferring with the environment. Otherwise we call the node e-incorrect. Similarly we call a node e-communicating if, on the challenge bit e, the prover sends some communication to the environment before giving a response.

Now let us look at the paths in the tree T that correspond to winning executions. For any node N along such a path, let e be the challenge bit that corresponds to the outgoing edge of N which lies on the path of the winning execution and let $\bar{e} = 1 - e$. Then

1. N is e-correct. This has to be the case since the path is accepting.
2. N is $\bar{e}$-incorrect or is $\bar{e}$-communicating. This has to be the case since otherwise the extractor would be able to extract a witness from this execution.

Now assume that two winning paths diverge from a node N. Then by property 1, N is 0-correct and 1-correct. By property 2, it then follows that N is 0-communicating and 1-communicating. But there can be at most ℓ such nodes on any path since the prover can communicate at most ℓ times. This shows that the (non-regular) subtree of T containing only winning paths contains at most 2^ℓ paths. There are $2^{\kappa+\ell}$ total paths in T and hence the probability of choosing a winning path is upper bounded by $1/2^\kappa$. We note that the above bound holds for any tree T and hence the probability of a bad execution occurring in a tree randomly chosen using the coins of P^* and $\mathcal{Z}$ is also upper bounded by $1/2^\kappa$ which is negligible in κ.

We note that in the above proof, we only need to limit communication *from* the environment *to* the prover. In other words, we actually described an (ℓ, ∞)-IPoK. In addition, the proof also works if we only restrict the *number of messages* from the environment to the prover, but allow each message to contain arbitrary many bits of information.

5.2 A Constant Overhead, Black-Box ZK IPoK for NP

We now describe a black-box ZK IPoK which has a constant overhead. For any relation R, let Σ be a Σ-protocol for R with conversations (x, a, e, z). We use Σ as a building block from which we compile our ℓ-IPoK protocol. As before, we use many repetitions of a Σ protocol with 1 bit challenges. However, the prover does not respond with the full value of z on each round, but only with a small share of z in some ramp secret sharing scheme. This way, the number of bits exchanged on each round is small. At the same time, if there are enough rounds on which the prover cannot communicate with the environment, the extractor can use rewinding and learn enough of the shares to recover the alternative response z' and hence the witness w.

The protocol uses a perfectly binding, computationally hiding commitment scheme which can commit to m bits using a $\mathcal{O}(m)$-bit string. It also uses a family of secret-sharing schemes SSS over some finite field $\mathrm{GF}(2^v)$. We write a secret sharing of a message z as $(Z[1], \ldots, Z[N]) = \mathrm{SSS}(z; r)$, where r is the randomness used. Here, $Z[i]$ are the shares and they are elements in the field $\mathrm{GF}(2^v)$. We call $(Z[1], \ldots, Z[N])$ a codeword. Using this notation, the full protocol is described in Fig. 9.

We assume that there exists a constant $\alpha > 0$ such that for any N there is an instantiation of the secret sharing scheme which shares a message consisting of αN field elements and has a privacy threshold αN (any αN shares of the codeword reveal no information about the shared secret). In addition, the sharing allows efficient reconstruction when any αN of the shares $Z[i]$ are lost (i.e. replaced by $\perp$). We call α the rate of the secret-sharing scheme. We could have postulated different rates for all the parameters, but since we only need them to be constant, working with one common rate greatly simplifies notation. Schemes of this type, where there is a large gap between the number of shares that maintain privacy and the number of shares needed to reconstruct, are often called "ramp schemes" and have the advantage that share size can be made significantly smaller than the size of the secret.

A well known ramp scheme can be constructed by modifying Shamir secret sharing so that the shares are defined by evaluating a polynomial of degree $2\alpha N - 1$ in which the secret makes up the top αN high degree coefficients and the remaining coefficients are random. This scheme has αN privacy and $2\alpha N$ shares can reconstruct so we just need $N \geq 3\alpha N$ or equivalently $\alpha \approx 1/3$. To get codewords of length N, we also need to use a field with $v \geq \log_2(N)$ which (as we will see later) will not give us a constant round scheme. However, it is also possible to use ramp secret sharing schemes over small (constant sized) finite fields. Such schemes were studied in [CC06], [CCGHV07].

The values of M and N depend on the security parameter κ and the communication threshold ℓ:

- The input to the prover is $(x, w) \in R$, and the verifier gets x.
- The following interaction is repeated for $m = 1, \ldots, M$:
  1. We first have a *commit phase*. The prover computes:
     (a) A random first message a_m for Σ.
     (b) A response $z_m^{(e)}$ to first message a_m and challenge e for $e \in \{0, 1\}$.
     (c) A secret sharing $Z_m^{(e)} = \mathrm{SSS}(z_m^{(e)}; r_m^{(e)})$ of the secret $z_m^{(e)}$ using randomness $r_m^{(e)}$.
     (d) A commitment $c_m^{(e)}$ to the pair $(z_m^{(e)}, r_m^{(e)})$.
     The prover sends $(a_m, c_m^{(0)}, c_m^{(1)})$ to V.
  2. We now have a *read phase* of N rounds, where in each round $n = 1, \ldots, N$ the verifier is offered to read the n'th field element in one of the codewords $Z_m^{(0)}$ or $Z_m^{(1)}$. Formally, for $n = 1, \ldots, N$
     (a) V chooses a challenge $e \in \{0, 1, \perp\}$ with probability distribution $\Pr(0) = \Pr(1) = \alpha/2$, $\Pr(\perp) = (1 - \alpha)$ and sends the challenge to V.
     (b) If $e \neq \perp$, P sends the field element $Z_m^{(e)}[n]$ to V. Else it sends back $\perp$.
     If during this the verifier tries to read more than αN field elements in a single codeword $Z_m^{(e)}$, then the prover stops the protocol, and the verifier rejects.[a]
  3. Lastly, there is a *verification phase*, where the verifier is allowed to see the opening to one of $c_m^{(0)}$ or $c_m^{(1)}$ to check that during the read phase it got valid shares of a valid response:
     (a) V sends a uniformly random challenge $b \in \{0, 1\}$ to P.
     (b) P sends an opening of $c_m^{(e)}$ to V which then recovers $(z_m^{(e)}, r_m^{(e)})$.
     (c) V verifies that
        i. The shares of $z_m^{(e)}$ received during the read stage were calculated correctly from the sharing $Z_m^{(e)} = \mathrm{SSS}(z_m^{(e)}; r_m^{(e)})$.
        ii. The conversation $(x, a_m, b, z_m^{(e)})$ is an accepting conversation of Σ.

[a] But this happens only with negligible probability.

Fig. 9. The Constant-Overhead Protocol

In particular, the result of [CCGHV07] shows how to use algebraic geometric codes to get a scheme with rate $\alpha = \frac{5}{21}$ in the finite field $\mathrm{GF}(2^v)$ with $v = 6$. The code is based on the curves of García and Stichtenoth for which there are efficient constructions.[4]

To simplify analysis we assume that the communication complexity of the original Σ protocol is $f(\kappa)$. In the protocol we pick

$$N \approx \max(\alpha^{-1} f(\kappa)\ ,\ 4^{-1}\alpha^{-1}\ell/\kappa) \tag{1}$$

$$M \approx (\beta_L + \beta_F + 1)\kappa + 1 \tag{2}$$

where we define the constants

$$\beta_L \approx 16\alpha^{-1}\log_2(e)\ , \quad \beta_F \approx -1/\log_2(\alpha/4) \tag{3}$$

Then SSS allows us to share a message consisting of $\alpha N \geq f(\kappa)$ field elements, each of length v bits, which gives the capacity of at least $f(\kappa)$ bits and hence enough to share a response z of the protocol Σ.

[4] Unfortunately, such codes do not exist for all N. However, for any N there is an N' in the interval $N \leq N' \leq 8N$ for which we can construct such a code. We ignore this subtlety in further discussion since it means at most a small constant blowup of our parameters.

Communication Complexity The communication complexity of all the commit phases and all of the verification phases is $\mathcal{O}(Mf(\kappa))$ The communication complexity of a single read phase is simply $(v+2)(N)$ since it takes 2 bits to encode the challenge e and v bits to encode the response. The communication complexity of all the read phases is then $MN(v+2)$. Since $N \geq f(\kappa)$, the total communication complexity of the protocol is then $\mathcal{O}(MNv)$. Under the assumptions that $\ell \geq 4\kappa f(\kappa)$, equation (1) just becomes $N \geq 4^{-1}\alpha^{-1}\ell/\kappa$. Assuming, in addition, that v is constant, the communication complexity of the protocol simply becomes $\mathcal{O}(\ell)$ which means that the protocol has a constant overhead for large enough ℓ.

The round complexity of the protocol is $\mathcal{O}(MN)$ which, under the above assumptions on ℓ and v, is also $\mathcal{O}(\ell)$.

Completeness It is clear that and honest prover and an honest verifier generate an accepting conversation as long as the verifier does not try to read more than αN positions in the same codeword. The expected number of field elements an honest verifier reads in a particular codeword is $(\alpha/2)N$. Using the Chernoff bound, it is easy to see that the probability of reading more than αN elements in a single codeword is negligible in N and hence also in κ. Using union bound, we see that the probability of this happening for any one of the possible $2M$ codewords is still negligible in κ.

Extractor The strong knowledge soundness extractor $\mathcal{X}$ is outlined in Fig. 10.

1. The extractor is given x and black-box rewinding access to a prover P^* with some initial randomness ω that produced an accepting conversation with V. In addition, the extractor gets the transcript of the original accepting conversation. It then tries to reconstruct as much as possible of the $2M$ codewords $Z_m^{(e)}$.
2. For each $m = 1, \dots, M$ and each index $n = 1, \dots, N$, the extractor rewinds the prover to the state it was in right prior to step 2 (a) of the protocol. For each of the challenge bits $e = 0, 1$:
   - The extractor runs the prover P^* on the challenge bit e. It replays any communication from $\mathcal{Z}$ to P^* that was sent at this point during the run of the actual protocol.
   - If the prover P^* attempts to send a message to the environment before outputting a response, the extractor counts this share as a loss and sets $Z_m^{(e)}[n] := \bot$. Also, if the prover communicated with the environment during this round in the original run of the proof (no matter what the original challenge was), the extractor sets $Z_m^{(e)}[n] := \bot$.
   - Otherwise, the prover sends a response $s \in \mathrm{GF}(2^v)$ and the extractor sets $Z_m^{(e)}[n] := s$. This is the same response the verifier would have gotten on this challenge during the original run.
3. After this, each of the $2M$ codewords $Z_m^{(e)}$ are defined, with some shares containing the *loss symbol*: $Z_m^{(e)}[n] = \bot$.
4. For each m the extractor attempts to reconstruct $z_m^{(0)}$, $z_m^{(1)}$ using the codewords $Z_m^{(0)}$, $Z_m^{(1)}$ respectively. If it succeeds, and both $(x, a_m, 0, z_m^{(0)})$, $(x, a_m, 1, z_m^{(1)})$ are accepting conversations, then the extractor uses special knowledge soundness of Σ to compute the witness w. Otherwise (if the above does not happen for any m) the extractor fails.

Fig. 10. The Strong Knowledge Soundness Extractor $\mathcal{X}$

We analyze the probability that a prover P^* working with the environment $\mathcal{Z}$ succeeds in producing an accepting conversation with an honest verifier V, but the extractor $\mathcal{X}$ subsequently fails to recover a witness. We say that P^* and $\mathcal{Z}$ win the extraction game if the above event occurs. Assume that there exist $\mathcal{Z}$ and P^* which win the extraction game with probability p. This probability is taken over the random coins of $\mathcal{Z}$, P^* and V (but extraction is deterministic). Then there is some particular value of the coins of $\mathcal{Z}$ and P^* for which they win the extraction game with probability at least p. In other words, if there exists $\mathcal{Z}$ and P^* which win the extraction game with non-negligible probability, then there also exist deterministic $\mathcal{Z}$ and P^* which win the extraction game with non-negligible probability. For the sake of an easier analysis we will therefore assume that $\mathcal{Z}$ and P^* are deterministic. We define an execution E as a random conversation between P^* (acting together with $\mathcal{Z}$) and V where the randomness is taken over the random coins of V only. We define $\mathcal{E}$ to be the set of all possible executions. Since extraction is deterministic, an execution completely defines the extraction process and, in particular, completely determines weather the extractor succeeds.

We define some subsets of $\mathcal{E}$. Let the accepting executions $\mathcal{A}$ be the subset of executions in which V accepts, and let the bad executions $\mathcal{B}$ be the subset of executions on which the extractor fails to extract a witness. Then $\mathcal{A} \cap \mathcal{B}$ is the set of accepting executions in which the extractor fails to extract. It is therefore sufficient to prove the following theorem:

Theorem 5. *For any deterministic strategy of a prover and an environment, we have* $\Pr[\mathcal{A} \cap \mathcal{B}] \leq (3)2^{-k}$.

We start with some definitions. For a given execution $E \in \mathcal{E}$, an epoch m denotes the portion of the conversation that corresponds to steps 1 through 3 of the protocol for a particular choice of $m \in 1, \ldots, M$.

For a given epoch $m = 1 \ldots, M$, a challenge round $n = 1, \ldots, N$, and a bit $e \in \{0, 1\}$ we define the share $Z_m^{(e)}[n] \in \mathrm{GF}(2^v) \cup \{\perp\}$ to be the share recovered by the extractor. Together, these shares define the codeword $Z_m^{(e)}$ recovered by the extractor.

We call $Z_m^{(e)}$ a faulty codeword if one or more of the shares $Z_m^{(e)}[n]$ is not consistent with the secret sharing $\mathrm{SSS}(z_m^{(e)}; r_m^{(e)})$, where $(z_m^{(e)}, r_m^{(e)})$ is the value contained in the commitment $c_m^{(e)}$ sent by the prover in the original proof.[5] Since the commitment scheme is perfectly binding, this notion is well defined, even though the extractor does not know which shares are faulty. We call m a faulty epoch if either of the codewords $Z_m^{(0)}$, $Z_m^{(1)}$ is a faulty codeword.

We call $Z_m^{(e)}$ a lossy codeword if more than αN of the shares $Z_m^{(e)}[n]$ contain the value $\perp$. We call m a lossy epoch if either of the codewords $Z_m^{(0)}$, $Z_m^{(1)}$ is a lossy codeword.

We call an epoch m an invalid epoch if one of the commitments $c_m^{(e)}$ for $e = 0$ or $e = 1$,is a commitment to some $(z_m^{(e)}, r_m^{(e)})$ such that the conversation $(x, a_m, e, z_m^{(e)})$ is not a valid conversation in Σ.

We call an execution faulty if it has more than $(\beta_F)\kappa$ faulty epochs. We call an execution lossy if it has more than $(\beta_L)\kappa$ lossy epochs and we call an execution invalid if it has more than κ invalid epochs. We use $\mathcal{F}$, $\mathcal{L}$ and $\mathcal{I}$ to denote the set of faulty, lossy and invalid executions respectively.

Lemma 1. *A bad execution is either faulty, lossy or invalid. I.e.,* $\mathcal{B} \subseteq \mathcal{F} \cup \mathcal{L} \cup \mathcal{I}$.

Proof. We show that $\overline{\mathcal{F} \cup \mathcal{L} \cup \mathcal{I}} \subseteq \overline{\mathcal{B}}$, so assume that $E \in \mathcal{E}$ is neither faulty, lossy nor invalid. Then E contains at most $(\beta_F + \beta_L + 1)\kappa$ faulty lossy or invalid epochs and hence it also contains at least one epoch which is neither faulty nor lossy nor invalid. In this epoch both of the codewords $Z_m^{(e)}$ have at most αN loss symbols and no faulty shares. Using the properties of the secret sharing

[5] Technically, we also include some border cases in this notion, such as the prover aborting or responding with garbage on the challenge e. In these cases the extractor can just set the share to be 0 and we call it a faulty share as well.

scheme, the extractor then correctly recovers both $z_m^{(0)}$ and $z_m^{(1)}$ which correspond to the committed responses. Since the epoch is also not invalid, these responses form two accepting conversations $(x, a_m, 0, z_m^{(0)}), (x, a_m, 1, z_m^{(1)})$ and hence the extractor recovers a witness w using special knowledge soundness of the underlying Σ-protocol. This means that $E \in \overline{\mathcal{B}}$.

Using the above lemma, we see that

$$\Pr[\mathcal{A} \cap \mathcal{B}] \leq \Pr[\mathcal{A} \cap (\mathcal{F} \cup \mathcal{L} \cup \mathcal{I})] \leq \Pr[\mathcal{A} \cap \mathcal{F}] + \Pr[\mathcal{A} \cap \mathcal{L}] + \Pr[\mathcal{A} \cap \mathcal{I}]$$

In Appendix D, we show that the three probabilities on the right hand side of the above equation are each bounded by $2^{-\kappa}$ which completes the proof of Theorem 5. The proof only relies on restricting the number of bits *from* the prover *to* the environment. Also, as in the previous protocol, we only need to restrict the number of exchanged messages but can allow each message to contain arbitrary many bits of information.

The ZK Simulator We now show that the protocol is 0-IZK which also implies that it is WI. Here we simply modify the usual simulator for simulating many repetitions of a Σ protocol with 1-bit challenges. On each epoch m, the simulator uses the special HVZK property to produce a random conversation (a, e, z) for Σ where e is a random bit. It then, in addition, produces a random secret sharing $\mathrm{SSS}(z; r)$ and a commitment $c_m^{(e)}$ to (z, r). In addition it produces a commitment $c_m^{(1-e)}$ to some garbage value. It then sends $(a, c_m^{(0)}, c_m^{(1)})$ to V^*. The simulator and the verifier then run the read phase of the protocol in which the simulator responds with random field elements for the challenges $1 - e$ and with secret shares of $\mathrm{SSS}(z; r)$ for challenges e. Lastly, in the verification phase if the verifier V^* sends the challenge e then the simulator honestly opens $c_m^{(e)}$ and goes on to the next round. On the other hand, if V^* sends the challenge $1 - e$ then the simulator rewinds V^* to the beginning of the epoch and tries again.

This is an expected polynomial time simulation. It is indistinguishable from a real execution by the hiding property of the commitment scheme and the privacy property of the secret sharing scheme.

5.3 Constant-Round, Non-Blackbox IPoK and IZK for NP

Constant-Round, Non-Blackbox IZK for NP In this section we sketch a non-black-box construction of an ℓ-IZK proof for any relation in NP, based on standard assumptions. The construction does not yield a proof of knowledge but only guarantees soundness (i.e. a cheating prover will not be able to prove a false statement). However we will show how to use this as a building block to construct an ℓ-IZK ℓ-IPoK protocol in Section 5.3. The construction is highly inefficient, and thus primarily of theoretic interest. It is obtained by a straight-forward modification of Barak's non-black-box zero-knowledge proof.

We will give the protocol only for cheating verifiers for which the code can be described with some *fixed* polynomial number of bits κ^a and which uses only a *fixed* polynomial number of random bits κ^b and which has a running time that is bounded by a *fixed* polynomial κ^c . A similar protocol was given by Barak for the standard setting (corresponding to full isolation). As in [Bar01] our basic protocol uses only perfectly binding commitments. Generalizing our protocol to tolerate arbitrary PPT cheating verifiers V^* can be done exactly as in [Bar01], by additionally assuming the existence of collision resistent hash functions which are secure even for an adversary that runs in $\kappa^{\log \kappa}$ time.

Assume that we have a perfectly binding commitment scheme commit, which can commit to any polynomial number of bits. For fixed integers $a, b \in \mathbb{N}$, the protocol is given in Fig. 11.[6]

1. The prover computes $c_1 = \text{commit}(0^{(a+b)\kappa}; r_1)$ and sends c_1 to V.
2. The verifier selects a uniformly random string $r \leftarrow \{0,1\}^{\ell+\kappa}$ and sends it to P.
3. Then P gives a WI proof that *either* $(x, w) \in R$ *or* there exists $M \in \{0,1\}^{(a+b)\kappa}$, r_1, $comm \in \{0,1\}^{\ell}$, such that $c_1 = \text{commit}(M; r_1)$ and M is a description of TM $M : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell+\kappa}$ for which $M(c_1, comm) = r$ and M runs in at most κ^c steps.

Fig. 11. Constant round ℓ-IZK protocol

Soundness We prove that the protocol is sound. By the soundness of the proof given in step 3, either $(x, w) \in R$ or there is some $(M, r_1, comm)$ such that $c_1 = \text{commit}(M, r_1)$, $comm \in \{0,1\}^{\ell}$ and M is a description of TM $M : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell+\kappa}$ for which $M(c_1, comm) = r$. To see that the second case happens with negligible probability, note that c_1 fixes M, as commit is perfectly binding. So, $M : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell+\kappa}$ and c_1 were fixed before r was sent. So, before r was sent there existed at most 2^{ℓ} strings r' for which $r' = M(c_1, comm)$ for some $comm \in \{0,1\}^{\ell}$. Since $r \in \{0,1\}^{\kappa+\ell}$, r hits one of these strings r' with probability at most $2^{-\kappa}$.

Non-Blackbox Isolated Zero-Knowledge. We prove that for any PPT environment $\mathcal{Z}$ and any PPT verifier V^* that can be described with κ^a bits and which uses at most κ^b random bits, there exists an ℓ-IZK simulator $\mathcal{S}$. For notational convenience we assume that the communication between V^* and $\mathcal{Z}$ is of the form where V^* sends arbitrary messages to $\mathcal{Z}$ which then returns single bit replies. We do not restrict the number of bits sent by V^* to $\mathcal{Z}$, but require that at most ℓ bits are sent from $\mathcal{Z}$ to V^* and hence the restriction that $\mathcal{Z}$ sends only one bit replies can be made without loss of generality as V^* can send the same message to $\mathcal{Z}$ multiple times.

The simulator $\mathcal{S}$ is given V^* and x as input, where V^* is described using a κ^a-bit string. It then defines a TM M: First pick $r_{V^*} \in \{0,1\}^{\kappa^b}$ and hard-code V^* and r_{V^*} into M. On input $(c, comm)$, M proceeds as follows: It runs V^* with randomness r_{V^*} and input c as if coming from P. Whenever V^* sends a message intended for $\mathcal{Z}$, it ignores this message and inputs the next bit from $comm$ to V^* as if coming from $\mathcal{Z}$. When at some point V^* outputs $r \in \{0,1\}^{\kappa+\ell}$ intended for P, the machine M outputs r. This described a TM $M : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{\kappa+\ell}$. Since r_{V^*} is κ^b bits long and V^* can be described using κ^a bits, we can describe M using $\kappa^a + \kappa^b$ bits. The simulator $\mathcal{S}$ then computes $c_1 = \text{commit}(M; r_1)$. Then it runs V^* with the randomness r_{V^*} previously hard-coded into M and sends c_1 to P^*. Then $\mathcal{S}$ runs V^* with $\mathcal{Z}$. Whenever V^* sends a message intended for $\mathcal{Z}$, $\mathcal{S}$ sends it to $\mathcal{Z}$ to get back a bit, which it inputs to V^*. When at some point V^* outputs some string $r' \in \{0,1\}^{\kappa+\ell}$ intended for P, then $\mathcal{S}$ records the bits $comm$ so far sent from $\mathcal{Z}$ to V^*, and pad $comm$ to have length ℓ. Note that by definition of M we will have that $M(c_1, comm) = r'$ for the r' just sent by P^*. So, now $\mathcal{S}$ can give the proof in Step 3 using the witness $(\epsilon, M, r_1, comm)$. Using the fact that the or-proof is WI, the simulation is indistinguishable from a real execution.

[6] The relation outlined in step 3 is in NP only when the running time of M is restricted to k^c steps. However, when the verifier can be an arbitrary PPT ITM V^*, then we cannot restrict M to a pre-defined polynomial limit on its run-time and hence resulting relation is not in NP. The result of [Bar01] shows how to get around this by using a witness indistinguishable proof system for the class $\mathbf{Ntime}(n^{\log \log n})$.

From IZK to IPoK + IZK Given any ℓ-IZK proof for NP we can construct a proof for NP which is ℓ-IPoK and ℓ-IZK. The construction uses a perfectly binding commitment scheme and a dense public-key encryption scheme. For notational convenience we assume that all κ-bit strings are public keys. The protocol is described in Fig. 12 and is based on ideas similar to the ones used in [BL02]. The protocol only has simulation based knowledge soundness but *not* strong knowledge soundness.

1. The verifier sends $c = \text{commit}(v; r)$ to P for a uniformly random $v \in \{0,1\}^\kappa$ and a randomizer r.
2. The prover sends a uniformly random $p \in \{0,1\}^\kappa$ to V.
3. V sends v to P and gives an ℓ-IZK proof that there exists r such that $c = \text{commit}(v; r)$.
4. If the proof succeeds, then both parties define a public key $pk = p \oplus v$ and P sends $C = E_{pk}(w; s)$ to V.
5. Then P gives an ℓ-IZK proof that there exist (w, s) such that $C = E_{pk}(w; s)$ and $(x, w) \in R$.

Fig. 12. From ℓ-IZK proofs to ℓ-IPoK, ℓ-IZK protocols

Theorem 6. *Assuming the existence of a perfectly binding commitment scheme and an IND-CPA secure dense public-key encryption scheme, there exists a constant-round IPoK + IZK compiler for NP. The constructed IPoK only has simulation based knowledge soundness.*

Proof. We first describe the simulation based knowledge soundness extractor. The extractor runs an internal copy of P^* and interacts with it by pretending to be a verifier. In the first step, the extractor $\mathcal{X}$ lets $c = \text{commit}(v; r)$ for a random v. Then when the extractor receives p from P^*, it samples a uniformly random key pair (pk, sk), sends $v' = p \oplus pk$ and simulates a proof to P^* that there exists r such that $c = \text{commit}(v'; r)$. If P^* then gives an accepting proof, the extractor $\mathcal{X}$ outputs a judgement $J = \texttt{accept}$ and outputs $w = D_{sk}(C)$. In the above process, the extractor $\mathcal{X}$ interacts with an internal copy of P^* in a manner that is indistinguishable from the interaction with an honest verifier and hence the environment can distinguish the real world interaction from interacting with an extractor with at most negligible probability. In addition, since the proof given by P^* is sound, implies that $w = D_{sk}(C)$ will be a correct witness with all but negligible probability when $J = \texttt{accept}$.

We now show that the protocol is ℓ-IZK. The simulator $\mathcal{S}$ runs the first three steps honestly, and then sends $C = E_{pk}(\epsilon; s)$. Then it simulates the ℓ-IZK proof for step 5. To prove that this is simulation is indistinguishable we need to appeal to the IND-CPA security of E_{pk}. We do this through a series of games argument:

1. We let game 1 be the execution with an environment $\mathcal{Z}$, an honest prover P and a cheating verifier V^*.
2. We let game 2 proceed as game 1 but use a simulator to simulate the proof in step 5 of the protocol. In this game, the simulator is given a valid ciphertext $C = E_{pk}(w; s)$. The game is indistinguishable from game 1 by the zero knowledge property of the proof used in step 5.
3. We define game 3, which proceeds as game 2, except that now the ciphertext C is created as $C = E_{pk}(\epsilon; s)$ and the simulator for step 5 uses C as the instance.

Game 3 is the simulation and hence we are left to show that games 2 and 3 are indistinguishable. We appeal to the IND-CPA security of the encryption scheme and the security of the coin-flip protocol which decides pk. Intuitively, if the coin-flip is secure then pk is random and so the encryption of

w and ϵ should be indistinguishable. Formalizing this intuition takes some effort and is not very interesting as the same argument appears in [BL02] hence we omit it.

This gives us the following corollary:

Corollary 1. *Assuming the existence of a collision resistent hash-function which remains collision resistant against $\kappa^{\log \kappa}$ bounded adversaries, a perfectly binding commitment scheme and an IND-CPA secure dense public-key encryption scheme, there exists a constant-round IPoK + IZK compiler for any relations in NP.*

Resettable-IZK IPoK The non-black-box technique of Barak can also be used to allow for proofs of knowledge with *resettable* or even *stateless* provers. This strong notion of security seeks to preserve the isolated zero knowledge property even when the verifying party can "reset" the honest prover and force it to run many times while re-using the same randomness. In practice this is especially relevant to our setting where partial isolation might be achieved by putting the prover on a smart-card or a hardware token which can be easily reset (i.e. by shutting off its power supply) and which does not have its own fresh source of randomness or the ability to keep long-term state. The idea of resettable zero knowledge was first considered by [CGGM99] which gave a construction of resettable zero knowledge proofs of membership but dismissed the idea of resettable proofs of knowledge as impossible. The intuition behind this impossibility is that the verifier's power to "reset" the prover makes it as powerful as the extractor and hence a cheating verifier would be able to retrieve a witness. However, it was later realized in [BGGL01] that, when the extractor is allowed to use non-black-box techniques, then it regains an advantage over the cheating verifier. This latter work gave a construction of resettable zero knowledge proofs of knowledge. We show how to extend those ideas to construct a resettable-IZK IPoK. We purposefully make the following discussion relatively informal and avoid rigorous proofs and definitions since we do not wish to veer too far from the main topic of the paper. It is meant to serve as an outline from which formal statements and proofs can be extracted with relative ease.

First we show how to modify the construction in Fig. 11 of an IZK protocol so that soundness is preserved even if a cheating prover can reset the honest verifier. Let $\{f_s : \{0,1\}^* \rightarrow \{0,1\}^{|s|}\}$ be a family of pseudorandom functions. In the construction Fig. 11, the modified honest verifier V is simply seeded with a random $s \in \{0,1\}^m$ and computes its challenges in each stage by applying f_s to the transcript so far (m is chosen to be as large as the largest challenge that can be sent during the protocol). This results in a resettably-sound zero knowledge proof as shown in [BGGL01].

We now modify the construction in Fig. 12 to be an IPoK + resettable-IZK protocol (i.e. the IZK property is preserved even if the cheating verifier can reset the prover). This is accomplished by adding a preliminary step in which the verifier sends a commitment C to his random coins R using a perfectly biding, computationally hiding commitment scheme. Then, whenever the verifier sends any message m to the prover, it also proves (using a resettably sound IZK protocol) that the message was generated using the code of an honest verifier and using the randomness R contained in the commitment C. The prover, consists of two modules P_1, P_2. The module P_1 acts as a verifier for the resettably sound IZK proofs authenticating V's messages m (it is seeded with some randomness s_1 used as a key for a pseudorandom function as described above). The module P_2 only gets the messages m and ignores the IZK proofs. It acts just like the prover in Fig. 12. The prover contains a second seed s_2. It initializes the randomness used by P_2 by applying f_{s_2} to the commitment C received as the first message sent by the verifier. Essentially, the first message C sent by the verifier, completely determines what future messages m the verifier may send. Thus resetting or rewinding the prover is useless since the verifier cannot send any modified messages. Although the result of [BGGL01] uses a slightly different approach, it defines a general class of admissible protocols for which the above transformation generates resettable-ZK proof of knowledge. It is straight-forward to show that our protocol falls within that class and hence is a resettable-IZK IPoK.

5.4 Constant-Overhead, Constant-Round IPoK for NP using a Random Oracles

In Section 4.3 we showed that non-black-box techniques are needed to construct a constant-round IPoK compiler. We now present a very efficient constant round protocol using random oracles. Later we discuss how to instantiate the random oracle using a non-black-box assumption.

As before, let R be an NP-relation, and let Σ be a Σ-protocol for R. We describe a constant round protocol Σ^+, which is intended to be a witness indistinguishable IPoK for R, constructed from Σ. We assume an oracle H that takes inputs of size $3\kappa + \ell$ bits and outputs κ bits. The protocol is given in Fig. 13.

1. First V sends a uniformly random string r of length $\kappa + \ell$ bits to P.
2. Then P starts running κ instances of Σ. It sends the first messages $a_1, ..., a_\kappa$ to V. Then, for $i = 1, \ldots, \kappa$:
   The prover P computes $z_i^{(0)}, z_i^{(1)}$, where z_i^e is the response to the first message a_i and the challenge bit e in Σ. The prover chooses random strings $r_i^{(0)}, r_i^{(1)}$ of length κ and sets $(s_i^{(0)}, s_i^{(1)}) = (H(r, r_i^{(0)}, z_i^{(0)}), H(r, r_i^{(1)}, z_i^{(1)}))$. Lastly, the prover sends $(s_i^{(0)}, s_i^{(1)})$ to V.
3. V sends random challenge bits $e_1, ..., e_\kappa$ to P.
4. For $i = 1, \ldots, \kappa$, P sends $z_i^{(e_i)}, r_i^{(e_i)}$ to V. By calling H, V checks that $s_i^{(e_i)} = H(r, r_i^{(e_i)}, z_i^{(e_i)})$, and also that $(a_i, e_i, z_i^{(e_i)})$ is an accepting conversation for Σ.

Fig. 13. A WI IPoK from a Random Oracle

Theorem 7. *The proof system Σ^+ is ℓ-IPoK for R. The overhead is $\mathcal{O}(1)$ for large enough ℓ. In addition Σ^+ is WI if Σ is WI.*

Proof. As for the overhead, the communication is that of κ runs of the Σ-protocol (which is $\mathrm{poly}(\kappa)$) plus the sending of r, a total of $\ell + \mathrm{poly}(\kappa)$. This gives an overhead of $1 + \mathrm{poly}(\kappa)/\ell$ which is $\mathcal{O}(1)$ for a large enough ℓ. In fact, the overhead achieved is $1 + o(1)$. The protocol runs in 4 rounds.

The required extractor simply looks at all oracle calls made by P^* and tests if there exists two calls specifying inputs of form $(r, r_i^{(0)}, z_i^{(0)}), (r, r_i^{(1)}, z_i^{(1)})$ where the outputs were used by P^* to form a pair $(s_i^{(0)}, s_i^{(1)})$ and where V would accept both $z_i^{(0)}$ and $z_i^{(1)}$. If so, it computes the witness using the special soundness property of Σ, otherwise it gives up.

Since P^* can send at most ℓ bits to the environment, the environment has at least κ bits of uncertainty about r. Therefore all calls to H where r appears in the input must have been made by P^*, except with negligible probability. Furthermore, since oracle outputs are κ bits long, they cannot be guessed except with negligible probability. Hence, any value $s_i^{(e_i)}$ that is checked by V in stage 4 of the protocol, must have been generated by P^* calling H on an input $r, r_i^{(e_i)}, z_i^{(e_i)}$ that V would accept. We say that such an element $s_i^{(e_i)} = H(r, r_i^{(e_i)}, z_i^{(e_i)})$ generated by P^* calling H is well formed.

It follows that, except with negligible probability, the only way in which P^* can construct a set of pairs $\{(s_i^{(0)}, s_i^{(1)})\}$ that will make V accept and the extractor fail is if every pair $(s_i^{(0)}, s_i^{(1)})$ contains exactly 1 well formed element. But then V accepts with probability only $2^{-\kappa}$.

If the underlying Σ-protocol is witness indistinguishable, than so are polynomially many repetitions of the protocol run in parallel. The only additional information the cheating verifier gets here are the hashes $s_i^{(\bar{e_i})} = H(r, r_i^{(\bar{e_i})}, z_i^{(\bar{e_i})})$ where $\bar{e_i} = 1 - e_i$ is the bit which the verifier did not pick as a challenge in stage 3 of the protocol. However, these hashes look random (even if the verifier knows a witness w and can guess $z_i^{(\bar{e_i})}$) unless the verifier guesses $r_i^{(\bar{e_i})}$ which only happens with negligible probability. Hence the protocol is indeed WI.

We have stated the above result in the random oracle model for simplicity. But actually, we only use the oracle in a limited way. We do not need a "programmable" oracle, i.e., the technique where the security reduction gets to decide what the oracle should output. Therefore, using our protocol does represent progress, in that one could not use our oracle to instead set up a common reference string, which allows ∞-IPoK, ∞-IZK protocols, as this requires programmability.

We do use the fact that a random oracle outputs do not reveal information on the inputs. However, the prover adds his own randomness when computing the hash and hence we should be able to achieve this hiding property under standard assumptions. We rely on the random oracle model to ensure that an output cannot be computed in a distributed fashion between two parties, each having only some portion of the input (i.e. the cheating prover knowing r and the environment knowing $z_{i,e}$). We believe it should be possible to instantiate our oracle with a concrete function and a well defined non-black-box assumption (such as the knowledge of exponent assumption) rather than basing ourselves on a heuristic.

5.5 From IPoK + WI to IPoK + IZK

For theoretical interest we include the following construction of an IPoK + IZK from a WI IPoK. In practice, this is only useful if we are in a situation where both the prover and the verifier can be assumed to be isolated. The construction is based on the FLS paradigm [FLS99].

Theorem 8. *Assuming the existence a perfectly binding, computationally hiding commitment scheme, there exists an IPoK + IZK compiler for every relation in NP.*

Proof. Let R be any NP relation. The verifier sends two commitments $C_0 = \text{commit}(m_0; r_0)$ and $C_1 = \text{commit}(m_1; r_1)$ to κ-bit random elements m_0 and m_1 using randomizers r_0 and r_1 respectively. Then V gives a WI ℓ-IPoK of (m, r) such that $C_1 = \text{commit}(m; r)$ or $C_2 = \text{commit}(m; r)$. It selects which witness (m_1, r_1) or (m_2, r_2) to use uniformly at random. If the proof is accepting, then P gives a WI ℓ-IPoK of (m, r, w) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$ or $(x, w) \in R$.

To show that the protocol is ℓ-IZK, the simulator runs V^* through the conclusion of the first WI IPoK protocol. If the proof given by V^* is accepting then, since V^* is ℓ-isolated, the simulator can extract some (m, r) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$. Then the simulator runs the second WI IPoK using the witness (m, r, ϵ), and the ℓ-IZK property follows from the witness indistinguishability of this proof.

To show that the protocol is ℓ-IPoK, the extractor simply extracts a witness in the WI proof given by the prover to get some (m, r, w) such that $C_0 = \text{commit}(m; r)$ or $C_1 = \text{commit}(m; r)$ or $(x, w) \in R$. If the extractor extracts a witness (m, r) for C_0 or C_1 then, with probability close to $\frac{1}{2}$, this differs from the witness used in the first ℓ-IPoK (by witness indistinguishability) and hence the prover and extractor together break the hiding property of the commitment scheme. Hence, with all but negligible probability, the extractor recovers a witness w such that $(x, w) \in R$.

Using the constant overhead WI IPoK for NP described in Section 5.2, we get the following corollary.

Corollary 2. *If there exists a perfectly binding, computationally hiding commitment scheme, then every NP relation R has an IPoK + IZK compiler with a constant overhead.*

6 Applications of WI IPoK

6.1 Preventing "Man-in-the-Middle" Attacks on Identification Schemes

An identification scheme is an interactive protocol where one party acts as a prover to securely prove its identity to another party acting as a verifier. Each prover has a public key which is known to all others. The usual solution has the prover perform a witness hiding proof of knowledge of the corresponding secret key. A "man-in-the-middle" attack on an identification scheme involves a cheating party simultaneously acting as a verifier for party A and a prover for party B. By simply redirecting messages between A and B the adversary is able to claim A's identity and successfully convince the party B. A previous solution for preventing such attacks, outlined in [CD97] requires a PKI in a strong sense: all the verifiers must have a registered public key for which they are guaranteed to know the secret key. Each prover then customizes his proof to a specific verifier by proving knowledge of either his or the verifier's secret key in a witness indistinguishable fashion. The verifier is then unable to redirect the proof to another party. Apart from requiring a strong PKI, in practice this also requires that the prover checks the identity of the verifier that is being communicated with. For instance, if you use your mobile phone to do a proof of identity and get access to some resource R, the phone must display the identity of R, so you can verify that you actually meant to access R.

As an alternative solution, we propose using the physical assumption that the prover is ℓ-isolated from all parties aside from the verifier. In the introduction, we discussed some scenarios where this could be a reasonable assumption. The prover uses a witness indistinguishable ℓ-IPoK to prove knowledge of the witness for one of two hard problems (e.g. knowledge of the discrete log of one of two randomly chosen group elements). By standard arguments, such a proof is witness hiding. On the other hand, for any ℓ-isolated prover that successfully runs such a proof, we have an extractor that can extract a witness, so if any cheating prover can impersonate an honest player, this breaks the hardness of the underlying problem (e.g. discrete log). This solution only requires that the verifier knows the correct public key for the prover, and for this a standard PKI suffices. In addition, the responsibility of not being fooled by man-in-the-middle attacks now falls, not on the prover, but on the verifier who must ensure that any prover he is interacting with is properly isolated. This places the burden on the physical design of the apparatus and so is much less prone to human mistakes.

6.2 Setting Up a PKI for General UC MPC

It is known that general multiparty computation secure in the UC framework is not possible without an honest majority and without any additional setup assumptions [CKL03]. To remedy this, previous work used reasonable setup assumptions such as the presence of a common reference string (CRS) or the existence of a public key infrastructure (PKI) where players are guaranteed to know the secret key corresponding to their registered public key. Both of the above assumptions require a trusted third party to initialize the setup. It is desirable to eliminate (or at least reduce) the level of trust required. For example, if the PKI is initialized by having players give their secret key/public key pair to a certificate authority (CA) then even a CA controlled by an honest-but-curious party would break the security of the system. It is interesting to note that we could initialize the PKI by having players provide a proof using the ∞-IPoK, HVZK protocol in Fig. 8, and thus allow Certificate Authorities which are controlled by an honest-but-curious party (or alternatively, a CA that acts honestly but makes all its communications public). However, the protocol is trivially insecure if the CA is actively malicious.

We instead propose using the physical assumption that a player can be partially isolated during a portion of the computation. A variant of this setting was previously considered in [Katz07], which showed that one can implement arbitrary multiparty computation in the UC framework without any trusted third parties using tamper proof hardware tokens. In particular, such a token is assumed to be able to interact with another player while being completely isolated from its owner (and hence also the environment). With ℓ-IPoK protocols, we can weaken the physical setup and only require that a party can be partially isolated from the environment during a portion of the computation. The parties register public keys with each other and provide proofs of knowledge of the corresponding secret keys using an ℓ-IPoK protocol where the prover functionality is ℓ-isolated from the environment. In a companion paper [DNW07], we show that this setup can be used as basis for UC secure multiparty computation tolerating an arbitrary number of adaptive corruptions. Note that, in particular, those results show that the witness indistinguishability property of the registration proof is sufficient and zero-knowledge is not required. This is an essential point, as in most settings it is unreasonable to assume that *both* of the interacting parties are isolated from the environment and we showed that one cannot achieve ZK without isolating the verifier to some extent.

7 Future Directions

The most interesting future research would be to improve the efficiency of the constructions we gave. In particular, it would be nice to have a smaller constant overhead than what we achieve in Section 5.2. Perhaps one could even find a black-box construction with an overhead of $1 + o(1)$ or show that such constructions are impossible. In addition, it would be interesting to come up with a specific reasonable non-black-box assumption (along the lines of the *knowledge of exponent* assumption) under which one could prove the security of the protocol in Fig. 13 or some similar protocol which runs in a constant number of rounds and has an overhead of $1 + o(1)$.

References

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd Annual Symposium on Foundations of Computer Science*, pages 106-115. Las vegas, NV, USA, 14–17 October 2001 IEEE.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology - Crypto '92*, pages 390–420, Berlin, 1992. Springer-Verlag. Lecture Notes in Computer Science Volume 740.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser and Yehuda Lindell Resettably-Sound Zero-Knowledge and its Applications In *42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, 14–17 October 2001. IEEE.

[BL02] Boaz Barak and Yehuda Lindell. Strict Polynomial-time in Simulation and Extraction In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 494–503, Montreal, Quebec, Canada, 2002.

[CGGM99] Ran Canetti, Oded Goldreich, Shafi Goldwasser and Silvio Micali. Resettable Zero-Knowledge Cryptology ePrint Archive 1999/022.

[CD97] Ronald Cramer, Ivan Damgård. Fast and Secure Immunization Against Adaptive Man-in-the-Middle Impersonations In W. Fummy, editor, *Advances in Cryptology - EuroCrypt '97*, pages 75–87, Berlin, 1997. Springer-Verlag. Lecture Notes in Computer Science Volume 1233.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - Crypto '94*, pages 174–187, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology - EuroCrypt 2003*, pages 68–86, Berlin, 2003. Springer-Verlag. Lecture Notes in Computer Science Volume 2656.

[CC06] Hao Chen and Ronald Cramer. Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields In C. Dwork, editor, *Advances in Cryptology - Crypto 2006*, pages 521–536, Berlin, 2006. Springer-Verlag. Lecture Notes in Computer Science Volume 4117.

[CCGHV07] Hao Chen Ronald Cramer Shafi Goldwasser, Robbert de Haan and Vinod Vaikuntanathan. Secure Computation from Random Error Correcting Codes In M. Naor, editor, *Advances in Cryptology - EuroCrypt 2007*, pages 291–310, Berlin, 2007. Springer-Verlag. Lecture Notes in Computer Science Volume 4515.

[DNW07] Ivan Damgård, Jesper Buus Nielsen and Daniel Wichs. Universally Composable Multiparty Computation with Partially Isolated Parties. Cryptology ePrint Archive 2007/332.

[FLS99] Uriel Feige, Dror Lapidot and Adi Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. In *SIAM Journal on Computing*, Volume 29, Issue 1, pages 1–28. Philadelphia, 1999. Society for Industrial and Applied Mathematics.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, 6–8 May 1985.

[Katz07] Jonathan Katz. Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In *Proceedings of EuroCrypt 2007*, pages 115-128, Springer Verlag LNCS 4515.

A A Computational Σ-Protocol for any PPT Relation

By a perfectly binding and computationally hiding commitment scheme we mean a function $\mathrm{commit}(m;r)$, where $\mathrm{commit}(m;r) = \mathrm{commit}(m';r')$ implies that $m = m'$, and where $\mathrm{commit}(m;r)$ and $\mathrm{commit}(m';r')$ are computationally indistinguishable when r and r' are uniformly random. Sometimes such schemes are defined in context of a key-generator $pk = \mathrm{gen}(s)$ and the commitment function defined by $\mathrm{commit}_{pk}(m;r)$; This is especially true when realized based on the RSA assumption or the DDH or DL assumption. In this case perfect binding is required for all $pk = \mathrm{gen}(s)$ and computational hiding required when s is uniformly random. The key and none-keyed notions are however equivalent, as the later form allows the first form. To commit, sample $pk = \mathrm{gen}(s)$ and $C = \mathrm{commit}_{pk}(m;r)$ and send $C' = (pk, C)$. To open, send (m, r, s) and let the receiver check that $C' = (\mathrm{gen}(s), \mathrm{commit}_{pk}(m;r))$. The following theorem has been used implicitly or explicitly in several places in the literature.

Theorem 9. *Assume that there exists a commitment scheme* commit*, which is perfectly binding and computationally hiding. Then there exists a computational Σ-protocol for all NP relations R.*

Proof. For any $(x, w) \in R$ we can use the NP reduction to Hamiltonian Path to compute, in PPT, a graph $G = G(x) = \{(i, j)\}$ on n nodes and a Hamiltonian path $P = P(x, w) = (i_1, \ldots, i_n)$ in $G(x)$. Furthermore, given $G = G(x)$ and any Hamiltonian path P in $G(x)$ one can in PPT compute w such that $(x, w) \in R$.

In the first message $P(x, w)$ use commit to commit to a random permutation, $\phi(G(x)) = \{(\phi(i), \phi(j)\}$, of $G(x)$: For all possible edges (i, j), if $(i, j) \in \phi(G(x))$, then send $C_{i,j} = \mathrm{commit}(1)$, otherwise, send $C_{i,j} = \mathrm{commit}(0)$. The message a consist of ϕ and the commitments $\{C_{i,j}\}$. Then V returns a one-bit challenge $e \in \{0, 1\}$. If $e = 0$ then P opens all commitments and sends the permutation ϕ. If $e = 1$ then P reveals the Hamiltonian path $(j_1, \ldots, j_n) = (\phi(i_1), \ldots, \phi(i_n))$ in $\phi(G(x))$ by sending $(j_1, \ldots, j_n)$ and opening $C_{j_1,j_2}, \ldots, C_{j_n,j_1}$. If $e = 0$ then V checks that ϕ is a permutation and that $C_{i,j}$ opens to 1 iff $(i, j) \in \phi(G(x))$. If $e = 1$ then V checks that $(j_1, \ldots, j_n)$ visits all nodes once and that $C_{j_1,j_2}, \ldots, C_{j_n,j_1}$ all opened to 1. This is clearly complete. As for special knowledge soundness, assume that for a fixed $a = (\phi, \{C_{i,j}\})$, the prover can reply accepting to $e = 0$ and $e = 1$. This gives an opening of $\{C_{i,j}\}$ to $\phi(G(x))$ plus an opening of $C_{j_1,j_2}, \ldots, C_{j_n,j_1}$ to 1. Since ϕ is a permutation and the commitment scheme is perfectly binding, this gives a Hamiltonian path $(\phi^{-1}(j_1), \ldots, \phi^{-1}(j_n))$ in $G(x)$, which in turn yields w such that $(x, w) \in R$. As for computational honest verifier zero-knowledge, assume that (x, e) is given. If $e = 0$, then S generates a and z honestly. If $e = 1$, then S let $C_{i,j} = \mathrm{commit}(1)$ for all (i, j), and picks a random Hamiltonian path $(j_1, \ldots, j_n)$ (the committed graph is complete) and opens $C_{j_1,j_2}, \ldots, C_{j_n,j_1}$ to 1. The only difference from the protocol is that when $e = 1$, then some of the commitments which are not opened contain 0 in the protocol but contain 1 in the simulation. This is indistinguishable by the computational hiding of $\mathrm{commit}(\cdot;\cdot)$.

B Σ-Protocols with Large Challenge Space are ℓ-IPoK for Very Small ℓ

We show that a Σ-protocol for a relation R is a expected $(\ell_{\mathcal{Z}}, \ell_P)$-IPoK as long as the channel between $\mathcal{Z}$ and P^* is trivial, in the sense that $\ell_{\mathcal{Z}} = c\log(\kappa)$ or $\ell_P = c\log(\kappa)$ for some constant c. We consider the cases $\ell_{\mathcal{Z}} = c\log(\kappa)$ and $\ell_P = c\log(\kappa)$ separately.

In both cases, P^* and $\mathcal{Z}$ are restricted from when P^* sends a until it sends z. Between these two points P^* and $\mathcal{Z}$ can communicate as they want, as long as entity X sends at most ℓ_X bits.

We can assume, without loss of generality, that all such communication takes place after P^* gets e and before it sends out z. Also without loss of generality, the transcript of the communication is an ordered sequence of messages $T = (m_P^{(1)}, m_{\mathcal{Z}}^{(1)}, \ldots, m_P^{(n)}, m_{\mathcal{Z}}^{(n)})$ where $m_X^{(i)}$ is the message sent by the party X and the number of messages is bounded by $n \le c\log(\kappa)$. We let $T_X = (m_X^{(1)}, \ldots, m_X^{(n)})$ denote the portion of the transcript that was sent by the party X.

We first consider the case $\ell_P = c\log(\kappa)$. Assume V accepts a conversation (x, a, e, z) for which the communication between P^* and $\mathcal{Z}$ is given by the transcript $(m_P^{(1)}, m_{\mathcal{Z}}^{(1)}, \ldots, m_P^{(n)}, m_{\mathcal{Z}}^{(n)})$. The extractor is outlined in Fig. 14.

1. The extractor $\mathcal{X}$ rewinds P^* to the point where it received e and inputs a new uniformly random challenge e'.
2. For $i = 1, \ldots, n$: The extractor $\mathcal{X}$ runs P^* until it sends $\hat{m}_P^{(i)}$ to $\mathcal{Z}$. If $\hat{m}_P^{(i)} \neq m_P^{(i)}$, then $\mathcal{X}$ aborts the attempt to run P^* on e' and starts again in step 1. Otherwise $\mathcal{X}$ gives the response $m_{\mathcal{Z}}^{(i)}$ to P^*.
3. The cheating prover P^* outputs z'. If (x, a, e', z') is not accepting, then $\mathcal{X}$ goes back to step 1. Otherwise $\mathcal{X}$ computes w using the special knowledge soundness property of Σ.

Fig. 14. Strong knowledge soundness extractor.

For fixed randomness $r_{\mathcal{Z}}$ of $\mathcal{Z}$ and fixed randomness r_P of P^*, each challenge e completely determines the communication between P^* and $\mathcal{Z}$, giving us a specific transcript $T(e)$. Since the random coins are fixed, the transcript $T(e)$ is completely determined by the messages sent by P^* to $\mathcal{Z}$ and hence $T_{P^*}(e)$ completely determines $T(e)$. The total length of the messages contained in $T_{P^*}(e)$ is $c\log(\kappa)$ and hence there are $2^{c\log(\kappa)} = \kappa^c$ possibilities for the bits that make up $T_{P^*}(e)$. For any such possibility, there are $2^{c\log(\kappa)} = \kappa^c$ ways to break up the bits into $n = c\log(\kappa)$ messages and hence the number of transcripts is bounded by κ^{2c}. This means we can divide the challenges $e \in \{0,1\}^\kappa$ into sets $S_T = \{e \in \{0,1\}^\kappa | T(e) = T\}$, where, on the challenge $e \in S_T$, the conversation between P^* and $\mathcal{Z}$ is given by the transcript T. We can partition each such set S_T into two disjoint sets R_T and A_T, where $e \in A_T$ makes P^* generate an accepting conversation and $e \in R_T$ makes P^* generate a rejecting conversation. Let $T_0 = (m_P^{(1)}, m_{\mathcal{Z}}^{(1)}, \ldots, m_P^{(n)}, m_{\mathcal{Z}}^{(n)})$ be the transcript given to the extractor at the conclusion of an accepting conversation between P^* and V. It can be seen that $\mathcal{X}$ is essentially sampling uniformly random challenges e' until it finds some $e' \in A_{T_0}$. Hence the expected running time of $\mathcal{X}$ is $2^\kappa / |A_{T_0}|$. This expectation is taken over the random coins of $\mathcal{X}$ alone. However, now taking the expectation over the random original challenge e produced by V, we want to compute the expected running time of $\mathcal{X}$ given that e is accepting. Let A be the set of challenges which produce accepting conversations. Let $T(e)$ be the transcript produced by the challenge e. Then the expected running time of $\mathcal{X}$ given that $e \in A$ is given by

$$\begin{aligned}
\sum_{e \in A} \frac{1}{|A|} \frac{2^\kappa}{|A_{T(e)}|} &= \frac{2^\kappa}{|A|} \sum_{e \in A} \frac{1}{|A_{T(e)}|} \\
&= \frac{2^\kappa}{|A|} \sum_{T} \left(\sum_{e \in A_T} \frac{1}{|A_T|} \right) \\
&\le \frac{2^\kappa}{|A|} \kappa^{2c} \qquad (4)
\end{aligned}$$

where (4) follows because the number of possible transcripts is bounded by κ^{2c}. The above expectation is taken for some set randomness of P^* and $\mathcal{Z}$. Each such randomness r produces a different set A_r of accepting challenges. Hence the overall expectation is

$$\sum_r \frac{2^\kappa}{|A_r|}\kappa^{2c}\Pr[r \mid r \text{ produced an accepting proof}] = \kappa^{2c}\sum_r \frac{2^\kappa}{|A_r|}\Pr[r]\frac{|A_r|}{2^\kappa}\frac{1}{\alpha} = \kappa^{2c}\frac{1}{\alpha}$$

where α is the overall success probability of P^* and $\mathcal{Z}$ producing an accepting conversation. We see that the above expectation is then polynomial in κ as long as α is non-negligible in κ.

Since the execution of P^* and $\mathcal{X}$ is expected poly-time and all e' are sampled uniformly at random, it follows that the probability that $e' = e$ is ever sampled is negligible. This in particular holds for the e' in the second accepting conversation (x, a, e', z') produced by $\mathcal{Z}$, meaning that $\mathcal{X}$ will be able to compute w with all but negligible probability. This shows the following lemma.

Lemma 2. *A Σ-protocol with large challenge space for R is a black-box, expected* $(\infty, \log(\kappa))$*-IPoK for R.*

We then consider the case where $\ell_{\mathcal{Z}} = c\log_2(\kappa)$ for some $c \in \mathbb{N}$. If V accepts a conversation (x, a, e, z), then $\mathcal{X}$ rewinds P^* to the point where e was input to P^* and inputs a new uniformly random e'. Then $\mathcal{X}$ tries all possible transcripts that $\mathcal{Z}$ can communicate with P^*. In other words, for each possible value of the $T_{\mathcal{Z}}$ (all possible communication from $\mathcal{Z}$ to P^*), the extractor $\mathcal{X}$ responds to P^* using the messages in $T_{\mathcal{Z}}$. As we saw before there are only κ^{2c} such possible transcripts. If in one of these runs P^* outputs z' such that (x, a, e', z') is acceptable, then $\mathcal{X}$ stops. Otherwise $\mathcal{X}$ picks a new e' and tries again. When a new acceptable (x, a, e', z') is generated, then $\mathcal{X}$ computes w if $e' \neq e$ and otherwise gives up.

For any randomness r of P^* and $\mathcal{Z}$ we can define the set of challenges A_r such that $e \in A_r$ iff P^* and $\mathcal{Z}$ produce an accepting conversation when given the challenge e. It is easy to see that expected running time is given by:

$$\sum_r \frac{2^\kappa}{|A_r|}\kappa^{2c}\Pr[r \mid r \text{ produced an accepting proof}] = \kappa^{2c}\frac{1}{\alpha}$$

where α is defined as in the proof of Lemma 2. In fact, the remainder of the proof is equivalent to that of Lemma 2 and hence we get the following:

Lemma 3. *A Σ-protocol with Large Challenge Space for R is a black-box, expected* $(\log(\kappa), \infty)$*-IPoK for R.*

The above lemmas show an *expected* black-box IPoK for any trivial channel between $\mathcal{Z}$ and P^*. In Theorem 3 we show that black-box extraction is impossible for any constant round protocol once the channel capacity becomes $\theta(\kappa)$.

We now argue that, there exists a Σ protocol which is not a black-box strict 0-IPoK.

Lemma 4. *If there exist perfectly binding, computationally hiding commitments, then a Σ-protocol with large challenge space for R need not be a black-box, strict* 0*-IPoK for R.*

We assume that there exists a semantic symmetric encryption where given one valid encryption $C = E_K(m)$ one cannot produce a new valid ciphertext $C' \neq C$, except with negligible probability.

We also assume that there exists a relation R and a computational Σ-protocol for R such that one can efficiently sample $(x, w) \in R$ such that the probability of computing w from x in PPT after seeing receiving a Σ-proof for x using witness w is negligible. Finally, we assume that there exist pseudo-random functions. All these assumptions are implied by the assumption that there exists perfectly binding, computationally hiding commitment.

We prove the lemma by giving a class of environments $\mathcal{Z}_{K'}$ indexed by K' and a class of cheating provers P_K^* such that no black-box extractor can extract all P_K^* in the context of all $\mathcal{Z}_{K'}$. In particular, any $\mathcal{X}$ will fail when $\mathcal{Z}_{K'}$ is chosen uniformly at random (i.e., K' is chosen uniformly at random) and P_K^* is chosen such that $K = K'$.

We first describe the environment $\mathcal{Z}_{K'}$. Its first action is to sample $(x, w) \in R$ using the hard sampler assumed above. Then it sends $E_{K'}(w, g, L, r)$ to P^*, where E is the encryption algorithm for the symmetric cryptosystem assumed above, $g \in_R \{0, 1, \ldots, \kappa\}$ is sampled uniformly at random, $L \in_R \{0, 1\}^\kappa$ is sampled uniformly at random and r is a uniformly random string for producing a first message $a = A(x, w; r)$ in the Σ-protocol for R.

The cheating prover P_K^* works as follows: It receives x as input. Then it waits for a message C from $\mathcal{Z}$. If C is not a valid ciphertext for key K, then P_K^* terminates. Otherwise, it computes $(w, g, L, r) = D_K(C)$. If $(x, w) \notin R$ or $g \notin \{0, 1, \ldots, \kappa\}$ or $L \notin \{0, 1\}^\kappa$ or r is not sufficiently long to act as randomness for computing the first message in the Σ-protocol, then P_K^* terminates. Otherwise, it uses the key L to define a pseudo-random function $F_L : \{0, 1\}^\kappa \to \{1, \ldots, 2^\kappa\}$, uses r and (x, w) to compute $a = A(x, w; r)$ and sends a to V. Then it receives e and responds with an acceptable z iff $F_L(e) \leq 2^{\kappa - g}$.

Assume that there exists an extractor which works for all PPT P^* and all PPT $\mathcal{X}$. Then it also works for P_K^* and $\mathcal{Z}_K$, where K is chosen uniformly at random.

When $\mathcal{X}$ is to extract a witness w from P_K^*, then it is given $C = E_K(w, g, L, r)$, and is given rewinding black-box access to P_K^*. Having rewinding black-box access to P_K^* gives $\mathcal{X}$ the following powers:

1. If $\mathcal{X}$ rewinds P_K^* to a point after it got input e and reruns it, then P_K^* will output the same z, which is useless.
2. If $\mathcal{X}$ rewinds P_K^* to a point after it sent a and before it got e as input, then $\mathcal{X}$ can supply a new challenge e', in which case it gets a new response z' iff $F_L(e') \leq 2^g$.
3. If $\mathcal{X}$ rewinds P_K^* to a point after it received C and before it sent a, then P_K^* sends the same a, reducing to the above case.
4. If $\mathcal{X}$ rewinds P_K^* to a point before it received C, then can send C^*. If it sends $C^* = C$, it reduces to the above case. If it sends $C^* \neq C$, then by the assumptions on the cryptosystem P_K^* will terminate, except with negligible probability.

By the semantic security of $C = E_K(w, g, L, r)$, it follows that we can assume that $\mathcal{X}$ received $C = E_K(0)$ instead. Therefore L can be assumed to be independent of the view of $\mathcal{X}$, meaning that we can replace F_L with a uniformly random function $\{0, 1\}^\kappa \to \{1, \ldots, 2^\kappa\}$ for the sake of argument. It can then be seen that the only way that $\mathcal{X}$ can use P_K^* for something useful is rewind it to the point where it sent a, and then rerun P_K^* on a new e' for which $F_L(e') < 2^g$. Now consider any $\mathcal{X}$ running in time κ^c, and let $d = c \log_2(\kappa)$ such that $2^d = \kappa^c$. If $d + 1 \leq g \leq d + 2$, then $\kappa^c \leq \frac{1}{2} 2^g$, meaning that $\mathcal{X}$ has time to run P_K^* on at most $\frac{1}{2} 2^g$ different e'. Since $\Pr[F_L(e') \leq 2^{\kappa - g}] = 2^{\kappa - g} / 2^\kappa = 2^{-g}$, independently for each $e' \neq e$, it follows that the probability that $\mathcal{X}$ runs P^* on $e' \neq e$ where $F_L(e') \leq 2^{\kappa - g}$ is at most $\frac{1}{2}$. This means that when $d + 1 \leq g \leq d + 2$ and the first conversation accepts, then $\mathcal{X}$ must with probability at least $\frac{1}{2}$ extract without getting any other useful information but the accepting conversation. The probability that $d+1 \le g \le d+2$ is at least $1/(\kappa+1)$ and the probability that the first conversation is accepting when $d+1 \le g \le d+2$ is $2^{\kappa-g}/2^{\kappa} = 2^{-g} \ge 2^{-d-2} = \frac{1}{4}\kappa^c$. So, with probability at least $\frac{1}{2}(\kappa+1)^{-1}\frac{1}{4}\kappa^c$ the extractor extracts an acceptable conversation without any further useful information, which is easily seen to contradict the assumed witness hiding property, as $\frac{1}{2}(\kappa+1)^{-1}\frac{1}{4}\kappa^c$ is polynomial in κ.

C Proof of Theorem 3

We start with any protocol having ρ rounds of communication and let $q = \lfloor \ell/\rho \rfloor$. Let $f : \{0,1\}^* \times \{0,1\}^m \to \{0,1\}^q$ be a pseudorandom function with keys of size m. The existence of pseudorandom functions follows from that of one way functions which are guaranteed to exist if witness hiding proofs of knowledge exist at all. We define a class of provers with (hardcoded) values $r, s \in \{0,1\}^m$ and $(x, w) \in \mathcal{R}$. For each such prover we have the corresponding environment with the (hardcoded) value r (which acts as a shared key between environment and prover) and the hardcoded instance x. A prover P^* and the corresponding environment $\mathcal{Z}$ are chosen randomly from this class.

Let us specify the interaction between P^*, $\mathcal{Z}$ and the verifier V. Essentially, P^* acts as the honest prover but checks in with the environment to make sure it has not been rewound. The interaction is outlined in Fig. 15.

The prover P^* begins by setting view to be the empty string. For $i = 1, \ldots, \rho$:

1. The verifier sends $v^{(i)}$ to P^*.
2. P^* sets $\mathsf{view} \leftarrow \mathsf{view}||v^{(i)}$, computes $\sigma_s^{(i)} \leftarrow f(\mathsf{view}; s)$, and sends $\sigma_s^{(i)}$ to $\mathcal{Z}$.
3. $\mathcal{Z}$ sends $\sigma_r^{(i)} \leftarrow f((\sigma_s^{(i)}, i); r)$ to P^*.
4. P^* verifies $\sigma_r^{(i)} = f((\sigma_s^{(i)}, i); r)$. If not then P^* quits. Otherwise P^* computes the response $p^{(i)}$ and sends it to V.

In the above interaction, $\mathcal{Z}$ has a counter to keep track of the round i. After it reaches $i = \rho$ and sends out $\sigma_r^{(\rho)}$, it aborts and stops responding to any incoming messages.

Fig. 15. Interaction between P^* and $\mathcal{Z}$ during proof with V

The outlined interaction has P^* send q bits on every round and receive q bits on every round. Since $q\rho \le \ell$, the cheating prover is indeed ℓ-isolated.

Assume that there is an extractor $\mathcal{X}$ which recovers a witness. Since the proof is witness hiding, the extractor must be able to get some some more output from P^*, other than just one run of the protocol (even if $\mathcal{X}$ acts as a dishonest verifier). However, the only way to do so in a black-box manner is to rewind P^* and get an additional response $p'^{(i)}$ for some round i. The only way this is possible is by $\mathcal{X}$ finding a collision on $f(\cdot; s)$ or guessing the value of $f(\cdot; r)$ on some point. This can be done in expected polynomial time if and only if $q = \mathcal{O}(\log(\kappa))$.

D Technical Lemmas for the Proof of Theorem 5

In this section we analyze the three probabilities $\Pr[\mathcal{A}\cap\mathcal{I}]$, $\Pr[\mathcal{A}\cap\mathcal{F}]$ and $\Pr[\mathcal{A}\cap\mathcal{L}]$ corresponding to invalid, faulty and lossy executions respectively. We show each of the probabilities is upper bounded by $2^{-\kappa}$.

Lemma 5. *The probability of an invalid accepting execution is upper bounded by* $\Pr[\mathcal{A} \cap \mathcal{I}] \le 2^{-\kappa}$

Proof. An invalid execution which is accepting has κ epochs in which one of the commitments is invalid (is for some message z' which is not a correct response in an accepting conversation of Σ). In each of these κ epochs, the verifier chose to open the commitment which is not invalid. To analyze the probability of such an even occurring we consider the following related game.

The game runs in M rounds and, in the beginning, your score is initialized to 0. At each round you have the option of getting a point by specifying a bit $b \in \{0, 1\}$. If you choose that option then a bit $\hat{b}$ is chosen randomly and you loose if $b = \hat{b}$. If you choose not to get a point then you just proceed to the next round. You only win if you manage to get κ points. It is clear that an optimal strategy in the above game is to choose to get a point on the first κ rounds but not afterwards. The winning probability of such a strategy is $2^{-\kappa}$.

It is easy to see the connection between the above game and a prover producing invalid executions. The bit b in the game corresponds to a prover committing to a response $z_m^{(e)}$ which does not lead to an accepting conversation. The bit $\hat{b}$ corresponds to the challenge chosen by the verifier. For an execution to be in $\mathcal{A} \cap \mathcal{I}$ it must be accepting and must contain at least k epochs where the prover chooses to take the risk of committing to an invalid response. This shows that a prover strategy which is able to produce an invalid yet accepting execution with probability p amounts to a strategy which can win the above game with probability p and hence $\Pr[\mathcal{A} \cap \mathcal{I}] \le 2^{-\kappa}$.

We will use this proof technique in the subsequent analysis as well. The games abstract away many of the details of the protocol and serve to simplify our proofs.

Lemma 6. *The probability of a faulty accepting execution is upper bounded by* $\Pr[\mathcal{A} \cap \mathcal{F}] \le 2^{-\kappa}$

Proof. For an execution to be in $\mathcal{A} \cap \mathcal{F}$, the prover has to make sure that there are $\beta_F \kappa$ epochs, each with at least one round n in which at least one of the challenges $e = 0$ or $e = 1$ gives an incorrect share. For each such epoch the verifier has at least $\alpha/2$ chance of choosing such a challenge e in round n. In the verification phase it then has an independent chance of a $1/2$ of asking for the opening to the commitment $c_m^{(e)}$. This means that for each such epoch the verifier will reject at the end of the epoch with probability $\alpha/4$.

Consider the following related game which runs in M rounds. At the beginning your score is initialized to $s = 0$. In each of the rounds $m \in 1, \ldots, M$ you can choose to collect a point in which case you score is updated to $s := s + 1$, but also a random experiment is performed in which you immediately loose the game with probability $\alpha/4$. If you choose not to collect a point the game goes on to the next round. You win if you collect $\beta_F \kappa$ points.

It is easy to see that the optimal strategy is to collect a point on the first $\beta_F \kappa$ rounds and not afterwards. The probability of winning the game is then upper bounded by $(\alpha/4)^{\beta_F \kappa} = (\alpha/4)^{(-1/\log(\alpha/4))\kappa} = 2^{-\kappa}$.

This shows that $\Pr[\mathcal{A} \cap \mathcal{F}] \le 2^{-\kappa}$.

Lemma 7. *The probability of a lossy accepting execution is upper bounded by* $\Pr[\mathcal{A} \cap \mathcal{L}] \le 2^{-\kappa}$

Proof. We prove the stronger statement that $\Pr[\mathcal{L}] \le 2^{-\kappa}$. In order for an execution to be in the set $\mathcal{L}$, there have to be $\beta_L \kappa$ epochs m in which at least one of the codewords $Z_m^{(e)}$ has at least αN loss symbols. This means that the execution has at least $\beta_L \kappa \alpha N$ rounds n in which the verifier communicates with the environment on at least one of the challenges $e \in \{0, 1, \perp\}$. For each such round there is a probability of at least $\alpha/2$ that the verifier chooses such an e and so the prover communicates during that round. However, the total number of communicated bits is always upper bounded by ℓ. Let $w = \beta_L \kappa \alpha N$

Consider the following game which runs in MN rounds. At the beginning you score is initialized to $s = 0$ and a penalty count is initialized to $c = 0$. In each round you can decide whether you want to collect a point or not. If you chose to collect a point, then we update your score to $s := s + 1$ but also a random experiment is performed in which your penalty count is updated to $c \leftarrow c + 1$ with probability *at least* $\alpha/2$. If you chose not to collect a point in some round, then s and c are left unchanged. You win the game if you make $s \geq w$ *and* $c \leq \ell$.

It is again clear that an optimal strategy is to collect a point in each of the w first rounds and then collect no points in the rest of the rounds.

The probability of getting a penalty point in each of the rounds where you collect a coin, is $\alpha/2$. So, the expected value of c is given by $C = (\alpha/2)w = \alpha^2/2\beta_L \kappa N$. Using equations (1), (3) we have

$$\ell \leq 4\alpha\kappa N = \alpha^2/4\beta_L \kappa N = 2^{-1}C$$

We use the Chernoff bound to conclude

$$\Pr[c \leq \ell] \leq \Pr[c \leq 2^{-1}C] \leq e^{-C/8} \leq e^{-\alpha^2 \beta_L \kappa N/16} \leq 2^{-k}$$

since $\beta_L = 16\alpha^{-1}\log(e)$ and $N \geq \alpha^{-1}$ This means that the success probability of the above game is bounded by 2^{-k} which in turn implies that $\Pr[\mathcal{A} \cap \mathcal{L}] \leq 2^{-\kappa}$.

Using the above three lemmas we see that $\Pr[\mathcal{A} \cap \mathcal{B}] \leq 3(2^{-\kappa})$ which is indeed negligible.