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 consider proof of knowledge protocols where the cheating prover may communicate with some external adversarial environment during the run of the proof. Without additional setup assumptions, no witness hiding protocol can securely ensure that the prover knows a witness in this scenario. This is because the prover may just be forwarding messages between the environment and the verifier while the environment performs all the necessary computation.

In this paper we consider an *ℓ-isolated* prover, which is restricted to exchanging at most ℓ bits of information with its environment. We introduce a new notion called ℓ-isolated proofs of knowledge (ℓ-IPoK). These protocols securely ensure that an ℓ-isolated prover knows the witness. To prevent the above-mentioned attack, an ℓ-IPoK protocol has to have communication complexity greater than ℓ. We show that for any relation in NP and any value ℓ, there is an ℓ-IPoK protocol for that relation. In addition, the communication complexity of such a protocol only needs to be larger than ℓ by a constant multiplicative factor.

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 helping the prover.

Nevertheless, in this paper we will consider a cheating prover who 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 (e.g., 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.

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 a parametrized IPoK compiler, or just IPoK, 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.

It turns out to be easy to construct an IPoK with overhead $O = \text{poly}(\kappa)$ 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. While this seems straightforward, it is not entirely trivial to prove that it works. Next, we show that, using novel techniques, it is also possible to construct an IPoK protocol with a *constant* overhead. This IPoK compiler generates protocols in which the number of rounds grows with the communication threshold ℓ. We show that this is necessary for any black-box extractable construction. However, using the non-programmable random oracle model as a non-black-box technique, we construct a constant round IPoK with an overhead that gets arbitrarily close to 1. Applying the non-black-box techniques introduced by Barak, we can get a constant-round construction based on a standard assumption. This last compiler, however, does not have a constant overhead. Our IPoK compilers are all non-trivial in that they produce protocols that are Zero Knowledge (ZK) in the standard sense (when the verifier is fully isolated), or at least witness indistinguishable (WI).

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. On the other hand, consider a cheating verifier that simply acts as a channel between the environment and the honest prover while the environment runs the honest verifier code to generate challenge messages. A simulator for this scenario must essentially run an accepting proof with the environment, which means that it must know a witness. Since the simulator is only given the instance and not the witness, this implies that ℓ-IZK proof of knowledge protocols with communication complexity $C \leq \ell$ only exist for trivial languages where the witness is easy to find. On the positive side, we show how to construct an ℓ-IZK, ℓ-IPoK protocol for any NP relation R and any pre-defined threshold 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. Secondly, in a followup paper [DNW07], we show how to implement arbitrary multiparty computation securely in the UC framework without relying on any trusted third parties if the players can be partially isolated during a short, initial proof phase. This improves on the work of [Katz07], which showed that arbitrary MPC is possible in the UC framework when parties are fully isolated by putting their functionality on a tamper-proof hardware token. In some sense, our follow-up work 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 $(x, w) \overset{?}{\in} R$ 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 $\texttt{accept}$. 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.

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.

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. We let some PPT environment $\mathcal{Z}$ pick $(x, w) \in R$ and then run (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. 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. 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 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 say that a protocol is an ∞-IPoK if 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 judgment $J \in \{\texttt{accept}, \texttt{reject}\}$.
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. Knowledge soundness extraction: $\textsc{Extr}^R_{P^*,V,\mathcal{Z},\mathcal{X}}(\kappa)$

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. 3.

To define zero-knowledge we compare the execution $\textsc{Exec}^R_{P,V^*,\mathcal{Z}}$ to a simulation $\textsc{Sim}^R_{\mathcal{S},\mathcal{Z}}$, 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. 4. 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 $|\Pr[\textsc{Sim}^R_{\mathcal{S},\mathcal{Z}}(\kappa) = 1] - \Pr[\textsc{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa) = 1]|$ 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 $\textsc{Exec} \in \{0, 1\}$.

Fig. 3. Execution $\textsc{Exec}^R_{P,V^*,\mathcal{Z}}(\kappa)$ 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 $\textsc{Sim} \in \{0, 1\}$.

Fig. 4. Simulation $\textsc{Sim}^R_{\mathcal{S},\mathcal{Z}}(\kappa)$

Isolation. The above definition of ∞-IZK, ∞-IPoK captures universally composable zero knowledge proofs of knowledge, as the cheating party is allowed arbitrary communication with its environment. We now describe how to model a corrupted party that 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 count the number of bits of information exchanged between $\mathcal{Z}$ and P^*. We say that P^* is $(\ell_{\mathcal{Z}}, \ell_P)$-isolated if P^* *sends* at most ℓ_P bits of information to $\mathcal{Z}$ and *receives* at most $\ell_{\mathcal{Z}}$ bits of information from $\mathcal{Z}$.

We restrict the cheating verifier in Fig. 3 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 $(\ell_{\mathcal{Z}}, \ell_V)$-IZK for R if we restrict the definition of zero knowledge to only $(\ell_{\mathcal{Z}}, \ell_V)$-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 parametrized IPoK, or just IPoK, which takes the security parameter κ and the isolation parameter ℓ as inputs and produces an ℓ-IPoK protocol. An IPoK + IZK compiler produces a protocol which is ℓ-IPoK and ℓ-IZK. 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.

4 Constructing IPoK Compilers

4.1 A Simple Construction

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 a 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 1. *The proof system Σ^* is an ℓ-IPoK for R. In addition, it is ZK in the standard sense of a fully isolated verifier.*

Proof. 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 Σ^* 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.

4.2 A Constant Overhead Construction

As before, let R be a relation in NP and let Σ be a Σ-protocol for R with conversations (a, e, z). We use Σ as a building block from which we compile our ℓ-IPoK protocol. We again use many repetitions of a Σ-protocol with 1 bit challenges. However, the prover does not respond with the full value of z in each round, but only with a small share of z in some ramp secret sharing scheme. This way, the number of bits exchanged in each round is small. At the same time, if there are enough rounds in 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.

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 $z_m^{(b)}$ to first message a_m and challenge b for $b \in \{0, 1\}$.
     (c) A secret sharing $Z_m^{(b)} = \mathrm{SSS}(z_m^{(b)}; r_m^{(b)})$ of the secret $z_m^{(b)}$ using randomness $r_m^{(b)}$.
     (d) A commitment $c_m^{(b)}$ to the pair $(z_m^{(b)}, r_m^{(b)})$.
     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 may 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 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.
  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 $c_m^{(b)}$ to V which then recovers $(z_m^{(b)}, r_m^{(b)})$.
     (c) V verifies that
        i. The shares of $z_m^{(b)}$ received during the read stage were calculated correctly from the sharing $Z_m^{(b)} = \mathrm{SSS}(z_m^{(b)}; r_m^{(b)})$.
        ii. The conversation $(x, a_m, b, z_m^{(b)})$ is an accepting conversation of Σ.

Fig. 5. The Constant-Overhead Protocol

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. Using this notation, the full protocol is described in Fig. 5.

Secret-sharing schemes with constant rate α, can be constructed using what is typically called "ramp schemes". 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 a rate of $\alpha \approx 1/3$ but requires us to use a field with $v \geq \log_2(N)$ which (as we will see later) will not give us a constant-overhead scheme. It is also possible to use a ramp scheme over a small (constant sized) finite field. Such schemes were studied recently in [CC06], [CCGHV07]. 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 field $\mathrm{GF}(2^v)$ with $v = 6$. The code is based on the curves of García and Stichtenoth [GS96] for which there are efficient constructions.[1]

Let $f(\kappa)$ be the communication complexity of the original Σ-protocol. In our construction we then 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

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

The scheme 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.

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 $\mathcal{O}(\ell)$.

[1] 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.

Completeness. It is clear that an 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.

Knowledge Soundness Extractor. The extractor tries to reconstruct as much as possible of the codewords $Z_m^{(0)}, Z_m^{(1)}$ for each $m = 1, 2, \ldots, M$. It rewinds to each round in which the prover did not communicate during the original execution of the protocol. The extractor then tries both of the challenges $0, 1$ and replays any communication from the environment to the prover that occurred in the original execution. If the prover tries to send out a message to the environment, the extractor gives up on recovering that share of the codeword and replaces it with a loss symbol $\perp$. Otherwise the extractor successfully recovers the same share that the verifier would have gotten during the real execution. At the end of this process, the extractor will hold some candidate codewords $\tilde{Z}_m^{(0)}, \tilde{Z}_m^{(1)}$ for each epoch $m \in 1, \ldots, M$. For each such m, either the extractor can correctly decode both codewords and recover the witness or:

1. One of the codewords has more than αN loss symbols.
2. One of the codewords contains at least one share which is faulty because it does not correspond to the committed secret sharing.
3. One of the shared secrets $z_m^{(b)}$ is an incorrect response to the first message a_m and the challenge bit b.

We show that the probability of an accepting conversation having one of the above possibilities occur in *each* epoch $m = 1, \ldots, M$ is negligible in the security parameter κ. Intuitively, there cannot be too many states in which the prover will communicate resulting in a loss symbols, since the prover is limited to sending at most ℓ bits to the environment. There also cannot be too many times where the prover sends faulty shares in an accepting conversation because such conversations have a high likelihood of the prover being "caught" in the verification phase. For the same reason, there cannot be too many times where the prover shares an incorrect response z_m. We formalize this intuition in the full version of this paper and show that the probability of the prover being involved in an accepting conversation with the verifier on which the extractor subsequently fails to extract a witness, is upper bounded by $3 \times 2^{-\kappa}$. 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 also ZK in the standard sense, 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. The simulator sends $(a, c_m^{(0)}, c_m^{(1)})$ to V^*. Then the simulator simulates the read phase of the protocol by responding with random field elements for the challenges $1 - e$ and with the secret shares of the codeword $\mathrm{SSS}(z; r)$ for challenges e. Lastly, in the verification phase, if the verifier V^* picks 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 and is indistinguishable from a real execution by the hiding property of the commitment scheme and the privacy of the secret sharing scheme.

4.3 Impossibility of a Constant Round Black-Box Extractable IPoK

In both of the IPoK constructions that we saw so far, the number of rounds grows with the communication threshold ℓ. Clearly, this has to be the case if we are only restricting the number of messages exchanged rather than the number of bits of information since otherwise the simple attack mentioned in the introduction will work. However, we now show that this is a necessary characteristic of any black-box extractable IPoK compiler even when we restrict the number of bits of information. In particular, once ℓ is super-logarithmic ($\ell = \omega(\log(\kappa))$), then no protocol with $\mathcal{O}(1)$ rounds can be a witness hiding ℓ-IPoK.

Theorem 2. *Any black-box construction of a witness hiding (expected) IPoK compiler, parametrized by the communication threshold ℓ and the security parameter κ, with ρ rounds of communication must satisfy $\ell/\rho = \mathcal{O}(\log(\kappa))$.*

Proof. We start with any protocol that runs in ρ rounds 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. The prover P^* acts just like an honest prover but checks in with the environment to make sure it has not been rewound prior to each round. The interaction is outlined in Fig. 6.

The outlined interaction has P^* send q bits on every round and receive q bits on every round. Since $q\rho \leq \ell$, 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 more output from P^*, other than just one run of the protocol. However, the only way to do so in a black-box

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

manner is to rewind P^* and get an additional response $p'^{(i)}$ for some round i. This is only possible if $\mathcal{X}$ finds a collision on $f(\cdot; s)$ or guesses the value of $f(\cdot; r)$ on some point, which can happen in expected polynomial time if and only if $q = \mathcal{O}(\log(\kappa))$.

4.4 Non Black-Box Techniques

In Theorem 2, we showed that non-black-box techniques are needed to construct a constant-round IPoK compiler. The idea of using non-black box techniques based on standard cryptographic assumptions was first studied by Barak in [Bar01]. In the full version of this paper, we show how to use Barak's techniques to construct a constant round IPoK + IZK compiler.[2] Since the extractor for such a protocol does not rely on rewinding, it is also possible to construct protocols that are resettable-ZK [CGGM99, BGGL01]: that is the zero knowledge property holds even when the verifier can reset the prover and force it to run multiple times with the same random coins. This is especially pertinent to our setting where isolation might be achieved by putting a prover on a smart-card which can be easily reset. Barak's non-black-box techniques are, however, inefficient in practice (requiring an application of the Cook-Levin reduction) and it does not seem that they can be used to get a constant round protocol which also has a constant overhead.

Here we take a different approach and present a very efficient constant round protocol using random oracles. As before, let R be an NP-relation, and let Σ be

[2] For the reader familiar with Barak's basic idea, the adaption to the isolated setting is fairly straight forward: The basic idea is that the prover produces a commitment c to some machine M, then the verifier returns a long random string r, and the prover shows that either the claim holds or $M(c) = r$. The simulator takes $M = V^*$ to be the cheating verifier. In the isolated setting we prove that either the claim holds or $M(c, aux) = r$ for some auxiliary input aux of length at most ℓ bits. The simulator takes aux to be the communication between V^* and $\mathcal{Z}$. By letting r be longer than $\ell + \kappa$ bits the soundness is maintained.

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. 7. A WI IPoK from a Random Oracle

a Σ-protocol for R. We assume an oracle H that takes inputs of size $3\kappa + \ell$ bits and outputs κ bits. The protocol is given in Fig. 7.

Theorem 3. *The proof system Σ^+ is ℓ-IPoK for R. In addition Σ^+ is WI if Σ is WI. The overhead of the given compiler is $1 + o(1)$ for large enough ℓ.*

Proof. The communication exchanged is that of κ runs of the Σ-protocol (which is poly(κ)) plus the randomness r and r_i and the tags s_i: a total of $\ell + \text{poly}(\kappa)$. This gives an overhead of $1 + \text{poly}(\kappa)/\ell$ which is $1 + o(1)$ for large enough ℓ. 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 Step 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 at most $2^{-\kappa}$.

If the underlying Σ-protocol is witness indistinguishable, then 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

Step 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. In reality, 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. 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 (along the lines of the knowledge of exponent assumption) rather than basing ourselves on a heuristic.

4.5 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 4. *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 $C_0 = \mathrm{commit}(m_0; r_0)$ and $C_1 = \mathrm{commit}(m_1; r_1)$ 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 $C_1 = \mathrm{commit}(m; r)$ or $C_2 = \mathrm{commit}(m; r)$. 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 $C_0 = \mathrm{commit}(m; r)$ or $C_1 = \mathrm{commit}(m; r)$ 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 $C_0 = \mathrm{commit}(m; r)$ or $C_1 = \mathrm{commit}(m; r)$. 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 $C_0 = \mathrm{commit}(m; r)$ or $C_1 = \mathrm{commit}(m; r)$ 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 $(x, w) \in R$.

5 Applications of WI IPoK

5.1 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 so that the verifier is 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 he communicates 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 simply uses a witness hiding ℓ-IPoK to prove his identity. The ℓ-IPoK property ensures that the prover himself knows a witness even if he simultaneously acts as a verifier in another instance of the proof, while the witness hiding property ensures that the verifier cannot learn such a witness. 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.

5.2 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 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. 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 by using tamper proof hardware tokens. In particular, it is assumed that a player can isolate such a token so that it cannot communicate even a single bit of information with any other party. 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 once and for all 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 followup 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.

6 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 4.2. 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. 7 or some similar protocol which runs in a constant number of rounds and has an overhead of $1 + o(1)$.

References

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, NV, USA, October 14–17, pp. 106–115. IEEE, Los Alamitos (2001)

[BG92] Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)

[BGGL01] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-Sound Zero-Knowledge and its Applications. In: 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, Nevada, October 14–17, IEEE, Los Alamitos (2001)

[CGGM99] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge Cryptology ePrint Archive 1999/022

[CD97] Cramer, R., Damgård, I.: Fast and Secure Immunization Against Adaptive Man-in-the-Middle Impersonations. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 75–87. Springer, Heidelberg (1997)

[CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

[CKL03] Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)

[CC06] Chen, H., Cramer, R.: Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)

[CCGHV07] Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure Computation from Random Error Correcting Codes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 291–310. Springer, Heidelberg (2007)

[DNW07] Damgård, I., Nielsen, J.B., Wichs, D.: Universally Composable Multiparty Computation with Partially Isolated Parties. Cryptology ePrint Archive 2007/332

[FLS99] Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions, Society for Industrial and Applied Mathematics. SIAM Journal on Computing 29(1), 1–28 (1999)

[GS96] García, A., Stichtenoth, H.: On the asymptotic behavior of some towers offunction fields over finite fields. J. Number Theory 61, 248–273 (1996)

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

[Katz07] Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)