On Probabilistic versus Deterministic Provers in the Definition of Proofs Of Knowledge

Mihir Bellare
Department Computer Science & Engineering
University of California at San Diego
9500 Gilman Drive, La Jolla, CA92093, USA.
mihir@cs.ucsd.edu.

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

October 22, 2006

Abstract

This note points out a gap between two natural formulations of the concept of a proof of knowledge, and shows that in all natural cases (e.g., NP-statements) this gap can be closed. The aforementioned formulations differ by whether they refer to (all possible) *probabilistic* or *deterministic* prover strategies. Unlike in the rest of cryptography, in the current context, the obvious transformation of probabilistic strategies to deterministic strategies does not seem to suffice *per se*.

Keywords: Proof of Knowledge, Probabilistic Proof Systems, Probabilism versus Determinism, Expected Running Time.

---

*Partially supported by the Israel Science Foundation (grant No. 460/05).

1 Introduction

The concept of a "proof of knowledge" was informally introduced by Goldwasser, Micali and Rackoff [4], and plays an important role in the design of cryptographic schemes and protocols (see, e.g., [2, 3]). This note refers to the common formulation of the aforementioned concept, which was given in [1].

Loosely speaking, the definition of a proof of knowledge requires the existence of a "knowledge extractor" that, when given access to any strategy, outputs the relevant information within (expected) time that inversely proportional to the probability that the given strategy convinces the knowledge verifier. Schematically, the definition of a proof of knowledge *requires something with respect to any strategy*.

The issue addressed in this note is the following. Usually, in definitions of the aforementioned type, it does not matter whether one quantifies over all probabilistic strategies or over all deterministic strategies. The reason is that, usually, satisfying the more restricted definition (which refers only to all deterministic strategies) immediately implies satisfying the general definition (which refers to all probabilistic strategies). Unfortunately, this does not seem to be the case in the current setting (of the definition of proofs of knowledge).

1.1 The source of trouble

In this subsection we provide a high-level description of the technical problem addressed in this work. We re-iterate this explanation, using more precise style after presenting the relevant definitions (in Section 2).

To clarify the source of trouble, let us first consider one of the usual settings (in which the problem does not arise): specifically, we consider the setting of zero-knowledge. In this case, the ability to simulate (in a black-box manner) any deterministic verifier strategy, implies the ability to simulate any probabilistic verifier strategy. The same holds also when we restrict attention to strategies that can be implemented by polynomial-size circuits. The reason is that given any probabilistic strategy, we may consider all residual deterministic strategies (obtained by all possible fixing of the strategy's coins), and obtain the desired simulation (for the probabilistic strategy) by combining all the corresponding simulations (i.e., of the residual deterministic strategies).

This simple argument (*per se*) fails when applied in the current context (of proofs of knowledge). Indeed, we can consider all residual deterministic prover strategies that emerge from a given probabilistic prover strategy, and we can combine the corresponding extraction procedures, but the combined procedure does not necessarily run in time that is inversely proportional to the probability that this prover convinces the verifier. For example, suppose that on input x, with probability $\frac{1}{2}$ (over the choice of the prover's coins), the residual prover convinces the verifier with probability $2^{-|x|}$ (where the probability here is over the verifier's moves), and otherwise the residual prover convinces the verifier with probability 1. Then, in the first case extraction may run in (expected) time related to $2^{|x|}$, whereas in the second case it runs for polynomial-time. It follows that the extraction for the original probabilistic prover strategy runs in (expected) time that is related to $\frac{1}{2} \cdot 2^{|x|}$. But this probabilistic prover strategy convinces the verifier with probability exceeding $\frac{1}{2}$. (Thus, this extractor does not run in time that is inversely proportional to the success probability of the probabilistic prover strategy.)

1.2 On the importance of relating the two definitions

Needless to say, when faced with two natural definitions we wish to know whether they are equivalent. Furthermore, we note that the two different definitions have appeared in the literature: For example, the definition in [1] refers to any probabilistic prover strategy, while the definition in [2, Sec. 4.7] only refers to (arbitrary) deterministic strategies (see further discussion in Section 2). Thus, equating the two definitions (which appear in two central texts on this subject) becomes even more important (as it aims at eliminating a source of confusion in the current literature).

In addition to the foregoing abstract motivation, there is also a concrete motivation to our study. It is typically easier to deal with deterministic strategies than with probabilistic ones, and thus relating the two definitions yields a useful methodology (i.e., demonstrating the "proof of knowledge" property with respect to deterministic strategies and deriving it for free with respect to probabilistic strategies). For example, we note that in [1, Apdx E] the "proof of knowledge" property (of the Graph Isomorphism protocol) is only demonstrated with respect to deterministic strategies, and this demonstration does not seem to extend to probabilistic strategies.[1]

Let us stress that in many applications the relevant prover strategies are in fact probabilistic. This is the case whenever proof-of-knowledge are the end goal (or close to it as in identification schemes), because in these cases the prover strategy represents an arbitrary adversarial behavior.[2]

2 Formal Setting

Let us start by recalling the definitional schema that underlies the two definitions that we study. Generalizing the treatment in [1] and [2, Sec. 4.7.1], we shall refer to an arbitrary class of potential (prover) strategies, denoted $\mathcal{S}$. Indeed, the treatment of [1] is obtained by letting $\mathcal{S}$ be the class of all (probabilistic) strategies, whereas the treatment of [2, Sec. 4.7.1] is obtained by letting $\mathcal{S}$ be the class of all *deterministic* strategies.

2.1 Preliminaries

Loosely speaking, deterministic strategies are functions that specify the next message to be sent by a party, based on its private input (which is hardwired in them) and as a function of the messages it has received so far. General (probabilistic) strategies are similar, except that the next message may also depend on a random input that is presented to these strategies. Formally, a (probabilistic) strategy σ is a function from $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^*$ such that $\sigma(\omega, \overline{\gamma})$ denotes the message to be sent by the corresponding party given that its random input equals ω, and the sequence of messages received so far equals $\overline{\gamma}$. Note that the strategy depends also on private inputs of the corresponding party, to which the outside world has no direct access. (These private inputs are hardwired in σ and do not appear explicitly in our notation.)

For a probabilistic strategy σ, we often consider residual deterministic strategies of the form $\sigma_\omega = \sigma(\omega)$ obtained by fixing the value of the random input to ω (i.e., $\sigma_\omega(\overline{\gamma}) = \sigma(\omega, \overline{\gamma})$).

[1] It seems that the authors of [1] overlooked this point. They either did not notice that the argument is restricted to deterministic strategies or assumed that the demonstration can be easily extended to probabilistic strategies. We note that the argument presented in [1, Apdx E] applies to any three-move Arthur-Merlin protocol for NP that has the following strong soundness property: given any two accepting transcripts (for the same input) that start with the same Merlin message but differ on Arthur's message, one can efficiently find a corresponding NP-witness.

[2] In contrast, in other applications, where proofs-of-knowledge are used as a tool (and the corresponding knowledge-extractor is used by some simulator), it suffices to consider deterministic prover strategies (because these are derived from residual deterministic strategies that are derived in the course of the security analysis).

Strategies will be used both as oracles and as specifying the actions of interactive machines. Specifically, we mean the following:

- When we discuss the interaction between parties on a common input, we incorporate this common input in each of the two strategies. The interaction of a strategy σ with a strategy σ' is the sequence of messages exchanged between the residual deterministic strategies σ_ω and $\sigma'_{\omega'}$, where ω and ω' are uniformly distributed. This sequence equals $\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots$ such that $\alpha_{i+1} = \sigma(\omega, (\beta_1, \ldots, \beta_i))$ and $\beta_i = \sigma'(\omega', (\alpha_1, \ldots, \alpha_i))$.
- When using σ as an oracle, the oracle machine may issue arbitrary queries, which need not be consistent with the way that σ interact with any interactive machine. In particular, these queries may relate to different values of random input ω, all chosen at the discretion of the oracle machine.

The second item represents a relaxation of the common interpretation of the definition of *using a probabilistic strategy as an oracle oracle*, and thus a short discussion is in place. The common interpretation of this notion is that the user (i.e., the oracle machines) is given oracle access to a (single) residual deterministic strategy (i.e., σ_ω) that is obtained from σ by fixing a uniformly distributed ω. In fact, all prior constructions of knowledge extractors used this interpretation. We believe, however, that the more liberal interpration suggested above (i.e., by which the user is given oracle access to σ itself) is consistent with the simulation paradigm and is adequate in all reasonable applications. Actually, the knowledge extractor constructed in this work refers to an intermediate interpretation (of using a probabilistic strategy σ as an oracle). By this interpretation the oracle machine may is given access to several residual deterministic strategies (i.e., several σ_ω's), which are derived from the same probabilistic strategy by the selection of independently and uniformly distributed values of the random input ω.

The relevant knoweledge. We capture the relevant knowledge by a binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ such that, on common input x, the "claimed knowledge" refers to knowledge of a string in $R(x) \stackrel{\text{def}}{=} \{y : (x,y) \in R\}$. The archetypical case is of NP-relations; that is, relations R that are polynomially bounded (i.e., $(x,y) \in R$ implies $|y| \leq \text{poly}(|x|)$) and are polynomial time recognizable (i.e., there exists a polynomial-time algorithm A such that $A(x,y) = 1$ if and only if $(x,y) \in R$). We denote by L_R the set of strings for which a "claim of knowledge" is not bluntly wrong; that is, $L_R \stackrel{\text{def}}{=} \{x : R(x) \neq \emptyset\}$.

2.2 The actual definitions

Our focus will be on the validity condition of the following definition, but for sake of completeness we state also the non-triviality condition.

Definition 1 (schema for defining proofs of knowledge): *Let R be a binary relation, and $\kappa : \{0,1\}^* \to [0,1]$. We say that an interactive machine V is a* knowledge verifier for the relation R with respect to a class of strategies $\mathcal{S}$ (and knowledge error κ) *if the following two conditions hold.*

Non-triviality: *For every $x \in L_R$, there exists a strategy $\sigma \in \mathcal{S}$ such that the verifier V always accepts when interacting with σ on common input x.*

Validity (with error κ): *There exists a probabilistic oracle machine K and a polynomial q such that, for every strategy $\sigma \in \mathcal{S}$ and every x, machine K satisfies the following condition:*

If when interacting with σ, on common input x, the verifier V accepts with probability $p_x > \kappa(x)$, then on input x when given oracle access to σ machine K outputs a string in $R(x)$ within an expected number of steps upper-bounded by

$$\frac{q(|x|)}{p_x - \kappa(x)} . \tag{1}$$

Note that the probability p_x depends on V, the strategy σ, and the common input x. The corresponding probability space is of all possible coin tosses of the strategies V and σ. Likewise, the probability space underlying Eq. (1) *consists of all possible coin tosses of the machine K and the strategy σ.*

The oracle machine K is called a (universal) knowledge extractor, *and κ is called the* knowledge error function.

In particular, it follows that $x \notin L_R$ implies $p_x \leq \kappa(x)$. We stress that, on input x and when given oracle access to a strategy σ that convinces V to accept x with probability exceeding $\kappa(x)$, the knowledge extractor always outputs a string in $R(x)$; that is, in this case, $\Pr[K^\sigma(x) \notin R(x)] = 0$. However, when the said probability does not exceed $\kappa(x)$, all bets are off. Nevertheless, if R is an NP-relation then we may assume, without loss of generality, that for every x and every σ it holds that $\Pr[K^\sigma(x) \notin R(x) \cup \{\perp\}] = 0$, where $\perp$ indicates halting without output. We now turn to the definitions studied in this note.

Definition 2 (the two definitions):

Following Definition 3.1 in [1]: *We say that V is a* knowledge verifier for the relation R with knowledge error κ *if Definition 1 holds with $\mathcal{S}$ being the set of all possible* (probabilistic) *strategies.*

Following Definition 4.7.2 in [2]: *We say that V is a* restricted knowledge verifier for the relation R with knowledge error κ *if Definition 1 holds with $\mathcal{S}$ being the set of all possible* deterministic *strategies.*

The two definitions differ only in the scope of strategies considered: [1, Def. 3.1] refers to all possible (probabilistic) strategies, whereas [2, Def. 4.7.2] refers only to all possible *deterministic* strategies.[3] Nevertheless, we show that in all natural cases (e.g., NP-relations) the restricted definition implies the general one.

2.3 Our result

Before stating this result formally, let us point out why it is not as obvious as analogous results regarding related definitions.[4] Suppose that V is a *restricted knowledge-verifier* (with knowledge error $\kappa = 0$) and let K be the corresponding knowledge extractor. Given a probabilistic strategy σ, the straightforward attempt to extract knowledge from σ consists of invoking K while giving it

[3] Unfortunately, these facts are not perfectly clear in the original texts: The formulation of [1, Def. 3.1] refers to all possible "interactive functions", yet the latter are defined in [1, Def. 2.1] as arbitrary *probabilistic* strategies. The formulation of [2, Def. 4.7.2] refers to all residual deterministic strategies that can be obtained by fixing the random input of some probabilistic strategy, but in retrospect the latter condition is a red herring (and does not help in extending this definition to the general case of [1, Def. 3.1]).

[4] Recall that simulation-security with respect to arbitrary (polynomial-size) deterministic adversaries typically implies simulation-security with respect to arbitrary probabilistic (polynomial-time) adversaries.

with oracle access to the residual deterministic strategy σ_ω, where ω is uniformly distributed. The problem is that the probability that σ_ω convinces V, denoted $p(\omega)$, may deviate arbitrarily from the probability that σ convinces V, denoted p. That is, the non-negative random variable $p(\omega)$ may behave arbitrarily subject (only) to the condition $p = \mathrm{E}_\omega[p(\omega)]$. This, in turn, implies that the expected running-time of K^{σ_ω} (taken also over the random choice of ω) is not necessarily inversely proportional to p. For example, it may be that $\Pr[p(\omega) = 2^{-n}] = 1/2$ and $\Pr_\omega[p(\omega) = 1] = 1/2$, and in this case the expected running-time of K^{σ_ω} may be 2^n while $\mathrm{E}_\omega[p(\omega)] > 1/2$.

Theorem 3 *Let V be a* restricted *knowledge verifier for R with knowledge error κ, where the length of the binary expansion of $\kappa(x)$ is polynomial in $|x|$. Suppose that the corresponding knowledge extractor, K, never outputs a wrong answer; that is, for every x and σ, it holds that $\Pr[K^\sigma(x) \notin R(x) \cup \{\perp\}] = 0$, where $\perp$ indicates halting without output. Then V is a knowledge verifier for R with knowledge error κ.*

Theorem 3 asserts that, under the additional assumptions regarding κ and K, the restricted definition (i.e., [2, Def. 4.7.2]) implies the general definition (i.e., [1, Def. 3.1]). As illustrated by the forgoing discussion, the corresponding knowledge extractor (for [1, Def. 3.1]) is not K (or the minor modification of K discussed above). We note that the two additional assumptions (regarding κ and K) can be easily met in case that R is an NP-relation. Details follows.

Recall that if R is an NP-relation then we can ckeck the output of K, and thus (on input x) we can always avoid outputting a string that is not in $R(x)$. This eliminates the additional assumption regarding K. As for the additional condition regarding κ, it can always be enforced by possiblly increasing κ a little; that is, by resetting $\kappa(x)$ to $\lceil 2^{q(|x|)} \cdot \kappa(x) \rceil / 2^{q(|x|)}$, where q is an arbitrary polynomial. Furthermore, in the case that R is an NP-relation, we may reset $\kappa(x)$ to $\kappa'(x) \stackrel{\text{def}}{=} \lfloor 2^{q(|x|)} \cdot \kappa(x) \rfloor / 2^{q(|x|)}$, for a sufficiently large polynomial q (by taking advantage of the fact that, for any $x \in L_R$, a string in $R(x)$ can be found in time $\exp(q(|x|))$).[5]

3 Proof of Theorem 3

Recall that the source of trouble is that for a uniformly distributed value of the random input, the success probability of the corresponding residual deterministic strategy (w.r.t convincing V) may be very different from the success probability of the original probabilistic strategy. This may lead to overwhelmingly long runs of the knowledge extractor (i.e., runs that contribute to the total expected running-time more than we can allow). The basic idea is to truncate such overwhelmingly long runs, and rely on the existence (in sufficient probability measure) of runs that are not overwhelmingly long.

Let us illustrate this idea by referring to the foregoing example, where $\Pr[p(\omega) = 2^{-n}] = 1/2$ and $\Pr[p(\omega) = 1] = 1/2$ (and $\kappa = 0$).[6] In this case, $p = \mathrm{E}_\omega[p(\omega)] > 1/2$, and so our extraction procedure should run in expected polynomial-time. Thus, we invoke K providing it with oracle

[5]This fact allows for handing the case that the probability that σ convinces V to accept x (i.e., p_x) is very close to $\kappa(x)$ in the sense that $p_x - \kappa'(x)$ is significantly larger than $p_x - \kappa(x)$. We first note that in this case $p_x < \kappa(x) + 2^{-q(|x|)}$ (as otherwise $p_x - \kappa(x) \geq 2^{-q(|x|)}$ and $p_x - \kappa'(x) < p_x - \kappa(x) + 2^{-q(|x|)} \leq 2 \cdot (p_x - \kappa(x))$). Thus, in this case (where $(p_x - \kappa(x))^{-1} < 2^{q(|x|)}$), we can afford running the standard search algorithm (which runs in time $2^{q(|x|)}$) in parallel to the given knowledge extractor. (We also use the fact that $p_x > \kappa'(x)$ implies $p_x > \kappa(x)$, which follows by the fact that (without loss of generality) the probability space underlying p_x is $\{0,1\}^{q(|x|)}$ (and thus $p_x > \kappa'(x)$ implies $p_x \geq \kappa'(x) + 2^{-q(|x|)} > \kappa(x)$).) On the other hand, if $p_x - \kappa'(x) = O(p_x - \kappa(x))$ then $(p_x - \kappa(x))^{-1} = O((p_x - \kappa'(x))^{-1})$. Thus, given an knowledge extractor of error κ, we obtain a knowledge extractor of error κ'.

[6]Throughout the text, n denotes the length of the common input x, which we often omit from the notation.

access to σ_ω, where ω is uniformly distributed among all possible random inputs, and truncate the execution after a polynomial number of steps has elapsed. If an output was obtained in this execution attempt then we output it, otherwise we repeat the experiment again. Note that, with probability 1/2, the residual strategy σ_ω satisfies $p(\omega)=1$, in which case K^{σ_ω} is expected to halt in polynomial-time with the desired output. Otherwise (i.e., $p(\omega)=2^{-n}$), the (truncated) execution of K^{σ_ω} may be useless, but it will not cause much harm (since it is suspended after a polynomial number of steps).

In the foregoing example we relied on a good *a priori* knowledge of the distribution of $p(\omega)$, which may not be available in general. Thus, in general, we shall employ a somewhat more sophisticated argument. Following is a rough sketch of the general argument, where we still assume for simplicity that $\kappa=0$. One key observation is that there exists an index i such that $\Pr_\omega[p(\omega)\approx 2^{-i}]$ is linearly related to $2^i\cdot p$ (where $p=\mathrm{E}_\omega[p(\omega)]$). We do not know this i and so we run, in parallel, numerous processes one per each of the relevant values of i. In the i^{th} process (i.e., the one related to the value i), we repeatedly attempt extraction with deterministic residual provers (derived by random fixings of ω), but truncate each attempt after $\mathrm{poly}(n)\cdot 2^i$ steps. Thus, for the correct value of i, the i^{th} relevant process will succeed in extraction within the allowed expected number of steps (i.e., it is expected to make $\mathrm{poly}(n)/(2^i\cdot p)$ attempts, each running for $\mathrm{poly}(n)\cdot 2^i$ steps, and thus the total expected running time is $\mathrm{poly}(n)/p$).

We now turn to a rigorous description of the actual knowledge extractor for probabilistic strategies. We fix an arbitrary $x\in L_R$, but omit it from most subsequent notations. Fixing an arbitrary randomized strategy σ, we consider an arbitrary choice of the strategy's coins, ω, and denote the residual strategy by σ_ω. In the rest, we will refer to selecting such ω's and providing oracle access to the corresponding σ_ω, but we need not select these ω's ourselves; it suffices to have the ability of providing oracle access to numerous random and independent "incarnations" of σ that correspond to such choices of ω's.

Let $p(\omega)$ denote the probability that verifier accepts when interacting with σ_ω, on common input x. By the hypothesis, if $p(\omega)>\kappa(x)$ then the knowledge extractor K, given oracle to σ_ω, outputs a string in $R(x)$ in expected time $q(|x|)/(p(\omega)-\kappa(x))$, where q is a fixed (universal) polynomial. As before, we let $p=E_\omega[p(\omega)]$, and assume, without loss of generality, that $p>\kappa(x)$ (because otherwise noting is required). In addition, let $\kappa=\kappa(x)$ and let $\ell=\mathrm{poly}(|x|)$ denote an upper-bound on *the length of the random input used by V on common input x*. It follows that for every choice of ω (which determines a residual strategy σ_ω) it holds that $2^\ell\cdot p(\omega)$ is an integer (because the relevant probability space is uniformly distributed over 2^ℓ possibilities). Recalling that κ has a binary expansion of length $\mathrm{poly}(|x|)$, we assume, without loss of generality, that $2^\ell\cdot\kappa$ is also an integer. It follows that if $p(\omega)\le\kappa+2^{-\ell-1}$ then $p(\omega)\le\kappa$.

We consider a partition of $(\kappa+2^{-\ell-1},\kappa+1]$ into $\ell+1$ intervals such that the i^{th} interval is $I_i=(\kappa+2^{-i},\kappa+2^{-i+1}]$. We claim that there exists $i\in[\ell+1]$ such that

$$\Pr_\omega[p(\omega)\in I_i]\ge\frac{2^i\cdot(p-\kappa)}{4(\ell+1)}\,. \tag{2}$$

This claim follows, because otherwise we derive a contradiction as follows (where in the first inequality we use the fact that $p(\omega)\le\kappa+2^{-\ell-1}$ implies $p(\omega)\le\kappa$):

$$\begin{aligned}E_\omega[p(\omega)] &\le \Pr_\omega[p(\omega)\le\kappa+2^{-\ell-1}]\cdot\kappa+\sum_{i=1}^{\ell+1}\Pr_\omega[p(\omega)\in I_i]\cdot(\kappa+2^{-i+1})\\ &= \kappa+\sum_{i=1}^{\ell+1}\Pr_\omega[p(\omega)\in I_i]\cdot 2^{-i+1}\end{aligned}$$

$$< \quad \kappa + \sum_{i=1}^{\ell+1} \frac{2^i \cdot (p-\kappa)}{4(\ell+1)} \cdot 2^{-i+1}$$
$$= \quad \kappa + \frac{p-\kappa}{2}$$

which, combined with the definition p (i.e., $p = E_\omega[p(\omega)]$) contradicts the hypothesis $p > \kappa$.

The new extraction procedure consists of running $\ell + 1$ processes in parallel. The i^{th} process successively invokes *time-bounded* executions of the knowledge extractor K, providing each such invocation with oracle access to a random and independent incarnation of σ (i.e., residual strategies σ_ω for uniformly and independently ditrsibuted values of ω). The time-bound used in the i^{th} process is $2 \cdot q(|x|) \cdot 2^i$, where the q is the polynomial guaranteed for K. Thus, if $p(\omega) \geq \kappa + 2^i$ then, with probability at least $1/2$, it holds that $K^{\sigma_\omega}(x)$ halts in $2 \cdot q(|x|) \cdot 2^i$ steps (because the expected number of steps is $q(|x|) \cdot 2^i$). Once any of these $\ell + 1$ processes outputs some string y, the entire parallel-process terminates and y is used as output.

Recall that by the theorem's hypothesis, whenever K outputs a string y it is the case that $y \in R(x)$. Thus, we confine ourselves to analyzing the expected running-time of the foregoing extraction process. Considering an arbitrary value i that satisfies Eq. (2), we observe that the i^{th} process succeed after making an expected number of $2 \cdot \left(\frac{2^i \cdot (p-\kappa)}{4(\ell+1)}\right)^{-1}$ trials. Thus, the overall time spent by the new extractor has expectation

$$(\ell+1) \cdot \frac{2 \cdot 4(\ell+1)}{2^i \cdot (p-\kappa)} \cdot (2 \cdot q(|x|) \cdot 2^i) = \frac{O(\ell^2 \cdot q(|x|))}{p-\kappa} = \frac{\text{poly}(|x|)}{p-\kappa}$$

and the theorem follows.

4 Concluding Remarks

We have established the equivalence of [1, Def. 3.1] and [2, Def. 4.7.2] while relying on the following three (reasonable) conventions (or assumptions):

1. We assumed that the pharse "given oracle access to a probabilistic strategy σ" means ability to query several (rather than one) residual deterministic strategies of the form σ_ω, where the ω's are uniformly and independently distributed.

2. We assumed that the knowledge-extractor never outputs a wrong string (i.e., a string not in $R(x)$), regardless of which input x and which strategy σ it is given access to.

3. We assumed that the knowledge error function κ is nice in the sense that, for every x, the binary expansion of $\kappa(x)$ has length polynomial in $|x|$.

We believe that these assumptions do not impair the applicability of our result. Still we wonder whether (some of) these assumptions can be eliminated.

References

[1] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 390–420.

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

[3] O. Goldreich. *Foundation of Cryptography – Basic Applications*. Cambridge University Press, 2004.

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