Appears in Proceedings of the 24th Annual ACM Symposium on the Theory of Computing (ACM, 1992).

# Making Zero-Knowledge Provers Efficient

(Extended Abstract)

Mihir Bellare* Erez Petrank†

**Abstract**

We look at the question of how powerful a prover must be to give a zero-knowledge proof.

We present the first unconditional bounds on the complexity of a statistical ZK prover. The result is that if a language possesses a statistical zero-knowledge then it also possesses a statistical zero-knowledge proof in which the prover runs in probabilistic, polynomial time with an NP oracle. Previously this was only known given the existence of one-way permutations.

Extending these techniques to protocols of knowledge complexity $k(n) > 0$, we derive bounds on the time complexity of languages of "small" knowledge complexity.

Underlying these results is a technique for efficiently generating an "almost" random element of a set $S \in \mathrm{P}$. Namely, we construct a probabilistic machine with an NP oracle which, on input $1^n$ and $\delta > 0$ runs in time polynomial in $n$ and $\lg \delta^{-1}$, and outputs a random string from a distribution within distance $\delta$ of the uniform distribution on $S \cap \{0,1\}^n$.

*High Performance Computing and Communications, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA. e-mail: mihir@watson.ibm.com

†Department of Computer Science, Technion, Haifa, Israel. e-mail : erez@techunix.technion.ac.il

# 1 Introduction

The investigation of the properties of interactive and zero-knowledge proofs [18] has already yielded many powerful results with diverse applications in complexity theory and cryptography. Here we continue this investigation by looking at the complexity of the prover in an interactive proof system, focusing in particular on the case where the proof system is statistical zero-knowledge. As we hope the following will indicate, bounding the complexity of zero-knowledge provers is not only interesting in its own right but also provides novel paradigms for some more classical complexity theoretic tasks.

Let us now proceed to describe our results.

## 1.1 Making ZK Provers Efficient without Assumptions

Suppose a language $L$ has an interactive proof $(P, V)$. The prover $P$ in such a proof is a (possibly probabilistic) function which given the common input and history of the interaction so far, returns the next message to send to the verifier $V$. The "complexity of the prover" refers to the computational complexity of this function.

The complexity of the prover has been addressed by Lund, Fortnow, Karloff and Nisan [22], Beigel, Bellare, Feigenbaum and Goldwasser [4] and (for multi-prover proofs [8]) by Babai, Fortnow and Lund [3]. It is also related to the notion of program checking of Blum and Kannan [9].

Here we are interested in the particular case where the interactive proof $(P, V)$ must also be statistical zero-knowledge[†] (statistical means that the zero-knowledge is in a strong sense: see §2 for definitions). One of the things that makes the question of the complexity of a statistical ZK prover intruiging is that the additional burden the ZK constraint puts on the prover appears to be quite large. In fact, no obvious upper bound on the power of a statistical ZK prover seems apparent. This is in contrast to the case of (plain) interactive proofs where a polynomial space prover always suffices. We note that this is true even though we know that languages in statistical ZK are of (relatively) low complexity: results of Fortnow [12] and Aiello-Håstad [1] imply that SZK $\subseteq$ AM $\cap$ co-AM $\subseteq \Sigma_2^P \cap \Pi_2^P$ (where SZK denotes the class of languages possessing statistical ZK interactive proofs of membership).

We do know that coins are necessary: Oren [23] and Goldreich and Oren [15] show that any ZK prover for a non-trivial language must be probabilistic.

Upper bounds on the complexity of a statistical ZK prover have been established by making use of (unproven) complexity assumptions. The first such result was that of Bellare, Micali and Ostrovsky [5] who showed that any language which possesses a statistical ZK proof also possesses a statistical ZK proof whose prover is a probabilistic, polynomial time (PPT) machine with a $\Sigma_2^P$ oracle, provided the discrete log problem is hard.[‡] Since then, both the complexity and the assumption have been reduced: on the one hand Ostrovsky [24] showed that the $\Sigma_2^P$ oracle may be replaced by an NP one, and on the other hand Ostrovsky, Venkatesan and Yung [25] show that any one-way permutation suffices.

To summarize, the cumulative effect of all this work was to establish that any language which possesses a statistical ZK proof also possesses a statistical ZK proof whose prover is a PPT machine with an NP oracle, given that one-way permutations exist. The question that remained was whether it was possible to do away with the cryptographic assumption entirely.

† The analogous question for computational ZK is easily resolved as a corollary of the results in [7, 20].

‡ [5] only claims a bound of probabilistic polynomial space, but the improvement to PPT with a $\Sigma_2^P$ oracle is immediate on combining their construction with the results of Jerrum, Valiant and Vazirani [21] on uniform generation.

Using different techniques, we answer this question in the affirmative. Namely, we establish the following unconditional result.

**Theorem 1.1** *Let $L$ be a language possessing a statistical zero-knowledge proof system. Then $L$ has a statistical zero-knowledge proof system in which the prover is a probabilistic, polynomial time machine with access to an NP oracle.*

This theorem is proved by a transformation. Given a statistical ZK proof system $(P, V)$ for $L$ we show how to construct a PPT oracle machine $P_{\mathit{eff}}$ such that $(P_{\mathit{eff}}^{NP}, V)$ is a statistical ZK proof system for $L$. Other than the unconditionality, our transformation offers some other improvements over previous ones. Some of these are

- We preserve the number of rounds ([5] blows them up by a constant factor and [25, 24] by a polynomial factor).
- We can preserve *perfect* ZK if we allow the prover to have a $\Sigma_2^P$ oracle instead of an NP one (in previous solutions, even if the original proof system had been perfect ZK, the transformed one would be statistical).

As in some of the results in [15, 23], one of the ideas in our proof is to make (appropriate) use of the "auxiliary" inputs that the definition of ZK provides (cf. [11, 15, 18, 23, 28]). Previous solutions [5, 24, 25] did not exploit this feature of ZK.

We note that auxiliary inputs are important to the definition of ZK: Goldreich and Krawczyk [13] show that without auxiliary inputs in the definition, ZK would not even be closed under sequential composition.

Another idea of our proof is to build a "universal" verifier which, for our purpose, captures the behaviors of all possible cheating verifiers. We believe that it is an interesting open question to determine whether some such universal verifier can effectively eliminate the need to consider all cheating verifiers in other ZK applications, and in particular be used to remove assumptions in other applications (cf. §6).

While previous solutions used only the fact that the given proof system $(P, V)$ was statistical ZK with respect to the honest verifier, our solution takes advantage of the full power of the ZK definition (that is, we use the fact that $(P, V)$ is ZK with respect to all verifiers).

We remark that Theorem 1.1 implies that SZK $\subseteq$ BPP$^{NP}$, providing an alternative way of showing that languages in SZK are of relatively low complexity (recall that the earlier and better results are due to Fortnow [12] and Aiello and Håstad [1]).

## 1.2 The Complexity of Non-Zero Knowledge Complexity

Knowledge complexity (KC) that is not zero was suggested by Goldwasser, Micali, and Rackoff [17] and recently redefined by Goldreich and Petrank [16]. The idea in the definition is that knowledge complexity measures the *computational* advantage gained in an interaction. The motivation to quantify KC originated from the hope that this new approach may shed new light on interesting questions in complexity theory. In this paper, we make a step towards understanding how the time complexity of a language depends on its knowledge complexity.

Our bounds on the time complexity of the language are obtained by bounding the complexity of the prover (using techniques similar to those used to establish Theorem 1.1). Thus, bounding the power of the prover becomes a tool for bounding the complexity of the underlying language.

Interestingly, our result involves a trade-off between the knowledge complexity and the number of rounds of the proof system. We are able to bound the time complexity of languages that have "short" interactive proofs of "small" knowledge complexity. More precisely, we show the following:

**Theorem 1.2** *If $L$ has an interactive proof of $g(n)$ rounds and of (statistical) knowledge complexity $k(n)$ satisfying $g(n)k(n) = O(\log n)$, then $L$ is in* $\mathrm{BPP}^{NP}$.

We do not know whether the restriction on the number of rounds is essential for this result or can be disposed of using better techniques, and we consider this question an interesting open problem.

We should stress one important difference between ZK proofs and proofs of higher KC. The difference is that, for the latter, one expects composition to increase the knowledge complexity. One consequence of this is that reducing the error probability by standard amplification techniques (such as serial composition) will increase the knowledge complexity commensurately. So, in saying what is a proof of knowledge complexity $k(n)$, it is important to say what the error-probability is. We adopt the natural notion that the error-probability is a negligible (i.e. less than the reciprocal of any polynomial) function of $n$, and that is what the above theorem assumes.

Our result can be viewed as an extension to higher KC of the results of [12, 1] showing that languages of zero (statistical) knowledge complexity (i.e. statistical ZK languages) have "low" complexity (they show that $\mathrm{SZK} \subseteq \mathrm{AM} \cap \text{co-AM} \subseteq \mathrm{BPP}^{NP}$). Our techniques, however, are quite different. Generalizing the techniques of [12, 1] to the setting of higher KC would yield a weaker result than Theorem 1.2 because it seems one must assume that the error-probability of the original proof system was exponentially small.[†]

## 1.3 Efficient Almost Uniform Generation

A technique underlying these results is to generate almost random elements of a set in probabilistic, polynomial time with an NP oracle. As this might be of independent interest, let us describe the result in more detail.

Let $S \subseteq \{0,1\}^*$ be a set decidable in polynomial time, and let $S_n = S \cap \{0,1\}^n$. The uniform generation problem is to generate, on input $1^n$, an element of $S_n$ distributed uniformly at random. Jerrum, Valiant and Vazirani [21], using results of Stockmeyer [27] on approximate counting, showed that uniform generation can be done in probabilistic, polynomial time with a $\Sigma_2^P$ oracle.

For our applications we would like a lower complexity than PPT with a $\Sigma_2^P$ oracle. On the other hand, we could tolerate a slight deviation from uniform of the output distribution. Accordingly, we consider the problem of *almost* uniform generation: on input $1^n$ and $\delta > 0$ generate a random element from a distribution within distance $\delta$ of the uniform distribution on $S_n$ (the distance between distributions $E_1$ and $E_2$ is defined as $\frac{1}{2}\sum_x |\Pr_{E_1}[x] - \Pr_{E_2}[x]|$).

If $\delta = n^{-c}$ for some fixed constant $c$ then techniques from Impagliazzo, Levin and Luby [19] can be used to do almost uniform generation in probabilistic, polynomial (in $n$) time with an NP oracle. However (for applications in this paper in particular) we would like to be able to achieve values of $\delta$ which are exponentially (in $n$) small, in the same complexity. We show that this can be done.

**Theorem 1.3** *Let $S \in \mathrm{P}$. Then there is a probabilistic oracle machine $A$ which on input $1^n$ and $\delta > 0$ runs in time polynomial in $n$ and $\lg \delta^{-1}$, and has the property that the distribution $A^{NP}(1^n, \delta)$ is within distance $\delta$ of the uniform distribution on $S_n$.*

In Theorem A.2 we actually prove something a little stronger: the almost uniform generation is "universal" (in the sense that $A$ does not depend on $S$ but rather gets a description of $S$ as an input).

[†] Extending Fortnow's techniques [12] one can show that if $L$ has a $g(n)$ round, KC $k(n)$, proof system with error probability $2^{-O(ng(n))}$, and if $g(n)k(n) = O(\log n)$, then $L \in \text{co-AM}$.

This result is established by combining techniques from Jerrum, Valiant and Vazirani [21] and Stockmeyer [27] with Carter-Wegman universal hash function [10] based techniques for estimating set sizes (Sipser [26]). The details are in Appendix A.

## 2 Preliminaries

If $E_1$ and $E_2$ are probability spaces then the distance between them, denoted $d(E_1, E_2)$, is

$$\frac{1}{2}\sum_{x\in\{0,1\}^*} |\Pr_{E_1}[x] - \Pr_{E_2}[x]| \ .$$

We say that $E_1$ and $E_2$ are $\delta$-close if the distance between them is $\leq \delta$.

An ensemble over $L \subseteq \{0,1\}^*$ is a collection $\{E(x,a)\}_{(x,a)\in L\times\{0,1\}^*}$ of probability spaces indexed by $L\times\{0,1\}^*$. We extend the definition of distance to ensembles, with the distance between ensembles $\mathcal{E}_1 = \{E_1(x,a)\}_{(x,a)\in L\times\{0,1\}^*}$ and $\mathcal{E}_2 = \{E_2(x,a)\}_{(x,a)\in L\times\{0,1\}^*}$ being the function $d^*\colon \mathsf{N} \to \mathsf{N}$ defined by

$$d^*(n) \ = \ \max_{x\in L, |x|=n} \ \sup_{a\in\{0,1\}^*} d(E_1(x,a), E_2(x,a)) \ .$$

A function $\epsilon\colon \mathsf{N} \to \mathsf{N}$ is negligible if for any constant $c > 0$ there is an integer $n_c$ such that $\epsilon(n) \leq n^{-c}$ for all $n \geq n_c$. We say that the ensembles $\mathcal{E}_1$ and $\mathcal{E}_2$ are *statistically indistinguishable* if the distance between them is negligible.

Next we define interactive proofs [18]. We begin by specifying the parties involved: the verifier and the prover. We have chosen to be very specific in our definitions, because this will simplify the exposition of our theorems and proofs.

Note that our verifiers take an auxiliary input (in addition to the common input and history of interaction) — intuitively, it captures a possible history of past interactions of the verifier.

In the following, $g, l, q\colon \mathsf{N} \to \mathsf{N}$ are polynomially bounded, polynomial time computable functions. We write $g, l, q$ for $g(n), l(n), q(n)$ respectively whenever $n$ is understood.

**Definition 2.1** *A* verifier *is a function $V$ with the following properties. Suppose $x \in \{0,1\}^n$, $\alpha_1, \beta_1, \ldots, \alpha_t, \beta_t$ are $l(n)$ bit strings, $a \in \{0,1\}^*$ and $R \in \{0,1\}^{q(n)}$. Then $V$ on input $x$ (the common input), $a$ (auxiliary input), $\alpha_1\beta_1\ldots\alpha_t\beta_t$ (the conversation so far), and $R$ (random tape of $V$) returns*

- *A $l(n)$ bit string $\alpha_{t+1}$ (next message to prover) if $t < g(n)$*
- *A bit (accept or reject) if $t = g(n)$.*

*Moreover $V$ runs in time polynomial in $n$. We refer to $g, l, q$ as the number of rounds, message length, and random tape length of $V$, respectively.*

Notice that by definition a verifier message must be a $l$ bit string, and a verifier must output some message for each round (it cannot halt in the middle of the protocol). Such conditions may be assumed without loss of generality anyway, and it will simplify our exposition and proofs to instead build them into the definition.

**Definition 2.2** *A* prover *is a (probabilistic) function $P$ with the following property. Suppose $x \in \{0,1\}^n$ and $\alpha_1, \beta_1, \ldots \alpha_{t-1}, \beta_{t-1}, \alpha_t$ are $l(n)$ bit strings, $t \leq g$. Then $P$ on input $x$ (the common input) and $\alpha_1\beta_1\ldots\alpha_{t-1}\beta_{t-1}\alpha_t$ (the conversation so far) returns a $l(n)$ bit string $\beta_t$ (next message to verifier). We refer to $g, l$ as the number of rounds and the message length of $P$, respectively.*

Again, note that a prover message in any round must be of length $l$. When we speak of a prover $P$ interacting with a verifier $V$, or couple them as a pair $(P, V)$, it is to be understood that both parties have the same number of rounds and same message length.

The probability that a verifier $V$ accepts (resp. rejects) at the end of an interaction with a prover $P$, on common input $x$ and auxiliary input $a$ for $V$, is the probability that $V(x, a, \alpha_1\beta_1 \ldots \alpha_g\beta_g, R) = 1$ (resp. 0) in the experiment given by choosing $R \in \{0,1\}^{q(n)}$ at random and setting $\alpha_t = V(x, a, \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}, R)$, $\beta_t = P(x, \alpha_1\beta_1 \ldots \alpha_t)$ for $t = 1, \ldots, g$ (here $g$ is the number of rounds, $l$ the message length, and $r$ the random tape length of $V$).

**Definition 2.3** *We say that a prover/verifier pair $(P, V)$ is an* interactive proof *for a language $L$ if*

(1) *The probability that $V$ rejects at the end of the interaction with $P$ on a common input $x \in L$ is negligible, and*

(2) *For any prover $\widehat{P}$ and common input $x \notin L$ the probability that $V$ accepts at the end of the interaction with $\widehat{P}$ is negligible*

*(the auxiliary input of $V$ is set to the empty string $\lambda$ in both cases). The first condition is called the* completeness *condition and the second the* soundness *condition.*

Next we define zero-knowledge (ZK) proofs [18]. We remark that the auxiliary inputs (cf. [11, 15, 18, 23, 28]) are crucial to make the definition meaningful. For one thing, without them, the composition of zero-knowledge protocols is not necessarily a zero-knowledge protocol as we would like it to be [13]. We recall also that any ZK proof which is black box simulation ZK (as all known ones are) is ZK in the auxiliary input model [15, 23].

Let us first define simulators. In the following let $p \colon \mathsf{N} \to \mathsf{N}$ be a polynomially bounded, polynomial time computable function.

**Definition 2.4** *A* simulator *is an algorithm $S$ which, on inputs $x \in \{0,1\}^n$, $a \in \{0,1\}^*$ and $r \in \{0,1\}^{p(n)}$, outputs, in time polynomial in $n$, a pair of strings $(R, \alpha_1\beta_1 \ldots \alpha_{g(n)}\beta_{g(n)})$, where $R \in \{0,1\}^{q(n)}$ and $\alpha_1, \beta_1, \ldots, \alpha_{g(n)}, \beta_{g(n)} \in \{0,1\}^{l(n)}$. We refer to $p$ as the number of coins the simulator tosses. We refer to $g, l, q$ as the number of rounds, message length, and random tape length of $S$, respectively.*

Note that we have not yet said anything about a simulator's ability to produce the view of the verifier. We do this in the next definition. For us, a simulator is simply an algorithm whose output format conforms to certain parameters. The purpose of this is again to simplify later exposition by building into the definition those features that could be assumed without loss of generality anyway.

For simplicity we are assuming a simulator always halts in polynomial time. We will indicate, as we go along, how our theorems change when we allow expected polynomial time simulators. Proofs for the latter case however are left to the final version.

If $V$ is a verifier and we talk of $S$ being a simulator for $V$ it is to be understood that these parties have the same number of rounds, message length, and random tape length. If $S$ is a simulator and $p$ the number of its coin tosses then we let $S(x, a)$ denote the probability space which to each string $s$ assigns the probability that $S(x, a, r) = s$ when $r \in \{0,1\}^{p(|x|)}$ is chosen at random.

Fix a (cheating) verifier $\widehat{V}$, common input $x$, and auxiliary input $a$ for $\widehat{V}$. We let $View_{(P,\widehat{V})}(x, a)$ denote the probability distribution on pairs $(R, \alpha_1\beta_1 \ldots \alpha_g\beta_g)$ which is given by picking at random $R \in \{0,1\}^{q(n)}$ and then setting $\alpha_t = V(x, a, \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}, R)$ and $\beta_t = P(x, \alpha_1\beta_1 \ldots \alpha_t)$ for $t = 1, \ldots, g$. Intuitively, this captures $\widehat{V}$'s view of the interaction: his own coin tosses and the transcript of the conversation with the prover.

**Definition 2.5** *Let $P$ be a prover, $\widehat{V}$ a verifier, and $S$ a simulator for $\widehat{V}$. We call $S$ a* statistical *(resp.* perfect*)* ZK $P$-simulator for $\widehat{V}$ *if the ensembles $\{\mathrm{View}_{(P,\widehat{V})}(x, a)\}_{(x,a) \in L \times \{0,1\}^*}$ and $\{S_{\widehat{V}}(x, a)\}_{(x,a) \in L \times \{0,1\}^*}$ are statistically indistinguishable (resp. equal).*

**Definition 2.6** *We say that* $(P,V)$ *is* statistical *(resp.* perfect*)* zero-knowledge *if for each verifier* $\widehat{V}$ *there is a statistical (resp. perfect) ZK* $P$*-simulator for* $\widehat{V}$*.*

We call $\widehat{V}$ a cheating verifier.

Finally we also summarize the definition of knowledge complexity which we use in the sequel. This definition is due to [16], and they call it the "fraction" version. For intuition and motivation we refer the reader to the original paper [16].

Let $S$ is a simulator, $p$ the number of its coin tosses, and $C \subseteq \{0,1\}^{p(n)}$ a subset of its coin tosses, then we let $S(x,a,C)$ denote the probability space which to each string $s$ assigns the probability that $S(x,a,r) = s$ when $r$ is chosen at random from $C$. In the following let $k\colon \mathsf{N} \to \mathsf{N}$.

**Definition 2.7** *Let* $P$ *be a prover,* $\widehat{V}$ *a verifier, and* $S_{\widehat{V}}$ *a simulator for* $\widehat{V}$*. Let* $p$ *be the number of coin tosses of* $S_{\widehat{V}}$*. We call* $S_{\widehat{V}}$ *a* statistical *(resp.* perfect*)* KC $k(n)$ $P$-simulator for $\widehat{V}$ *if for each* $n$ *and each* $x \in L \cap \{0,1\}^n$ *there is a set* $SUCC_{x,a} \subseteq \{0,1\}^{p(n)}$ *such that*

(1) $2^{-p(n)}|SUCC_{x,a}| \geq 2^{-k(n)}$ *(*$SUCC_{x,a}$ *has density at least* $2^{-k(n)}$*)*

(2) *The ensembles* $\{\mathrm{View}_{(P,\widehat{V})}(x,a)\}_{(x,a)\in L\times\{0,1\}^*}$ *and* $\{S_{\widehat{V}}(x,a,SUCC_{x,a})\}_{(x,a)\in L\times\{0,1\}^*}$ *are statistically indistinguishable (resp. equal).*

*We call* $SUCC_{x,a}$ *the* success set of $S$ for $x$.

**Definition 2.8** (Knowledge Complexity – Fraction Version) *We say that* $(P,V)$ *has* statistical *(resp.* perfect*)* knowledge complexity $k(n)$ *for a language* $L$ *if for any verifier* $\widehat{V}$ *there exists a statistical (resp. perfect) KC* $k(n)$ $P$*-simulator* $S_{\widehat{V}}$ *for* $\widehat{V}$*.*

PPT abbreviates "probabilistic, polynomial time."

## 3 The Simulation Based Prover

A basic tool in our results is the construction, from the simulator for a protocol, of a new prover based on this simulator. Here we describe a key component of this construction which we call a simulation based prover, and develop techniques to implement it efficiently. Similar ideas were used first in [5] and then later in [24]. These papers however did not incorporate auxiliary inputs into the notion, a difference that is important to some of our results.

**Definition 3.1** *If* $S$ *is a simulator of* $g$ *rounds and message length* $l$ *(cf. Definition 2.4) and* $S(x,a,r) = (R, \alpha_1\beta_1 \ldots \alpha_g\beta_g)$ *then for each* $t = 1,\ldots,g$ *we let* $S_t(x,a,r) = \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}\alpha_t$*.*

**Definition 3.2** *Let* $S$ *be a simulator (of* $g$ *rounds and message length* $l$*), and let* $p$ *be the number of its coin tosses. We denote by* $P_S$ *the probabilistic function whose output (next message to verifier), on input* $x$ *(common input),* $a$ *(auxiliary input) and* $\alpha_1\beta_1 \ldots \alpha_t$ *(conversation so far), is given by the following experiment:*

(1) *Pick at random a string* $r$ *from the set* $\{\, r \in \{0,1\}^{p(n)} : S_t(x,a,r) = \alpha_1\beta_1 \ldots \alpha_t \,\}$

(2) *Then compute* $S(x,a,r)$ *— this has the form*

$$(R, \alpha_1\beta_1 \ldots \alpha_t\beta'_t \ldots \alpha'_g\beta'_g)$$

*for some* $\beta'_t, \ldots, \alpha'_g, \beta'_g \in \{0,1\}^l$

(3) *Finally, output the string* $\beta'_t$*.*

*We call* $P_S$ *the* simulation based prover *for* $S$*.*

Note that $P_S$ is not quite a "prover" in the sense of Definition 2.2 because it has too many arguments: a prover does not take an auxiliary input. We choose to name it as we have because we will, later, use it as a basis to construct "true" provers.

Intuitively, if $S$ is a (statistical) ZK $P$-simulator for $V$, then $P_S$ is attempting to find $P$'s replies in the interaction of $P$ with $V$. He does this by using $S$ to appropriately sample the view of $V$ in its interaction with $P$. The role played by the auxiliary input is more subtle, and we will discuss it later. It should be noted, though, that in interacting with $P$ the verifier $V$ may have some auxiliary input of its own, and this is not necessarily the same as the auxiliary input to $P_S$.

We will denote the output of $P_S$ on input $x, a$ and $\alpha_1\beta_1 \dots \alpha_t$ by $P_S(x, a, \alpha_1\beta_1 \dots \alpha_t)$, but keep in mind the function is probabilistic so that $P_S(x, a, \alpha_1\beta_1 \dots \alpha_t)$ is actually a distribution on $l$ bit strings.

The complexity of computing $P_S$ is the question we address next.

Exact computation of $P_S$ is clearly possible in probabilistic polynomial space, and this is easily improved to PPT with a $\Sigma_2^P$ oracle by using the results of [21] on uniform generation.

Let us now turn to approximations: we will be interested in computing a distribution which is $\delta$-close to $P_S$. If $\delta = n^{-c}$ for some constant $c$ then, following [24], this can be done in PPT with an NP oracle by using techniques from [19]. That result will not, however, suffice for the applications in this paper: we need to be able to achieve values of $\delta$ which are exponentially small (in $n$). The following Theorem says we can do this, still in PPT with an NP oracle.

**Theorem 3.3** *Let $S$ be a simulator (of $g$ rounds and message length $l$). Then there exists a probabilistic oracle machine $T$ with the following property. Suppose $x \in \{0,1\}^n, a \in \{0,1\}^*$, $\alpha_1, \beta_1, \dots, \alpha_t$ are $l(n)$ bit strings ($t \leq g$), and $\delta > 0$. Then the probability spaces $T^{NP}(x, a, \alpha_1\beta_1 \dots \alpha_t, \delta)$ and $P_S(x, a, \alpha_1\beta_1 \dots \alpha_t)$ are $\delta$-close. Moreover, $T$ runs in time polynomial in $n$ and $\lg \delta^{-1}$.*

Note the running time of $T$ is polynomial in $\lg \delta^{-1}$ (rather than $\delta^{-1}$), which is why we can achieve exponentially small error.

The proof of this theorem relies on an implementation of *universal almost uniform generation* in PPT with an NP oracle. We leave these results to Appendix A. Given these results, the proof of Theorem 3.3 is straightforward: follow the algorithm of Definition 3.2, using Theorem A.2 to implement the first step.

# 4 Making ZK Provers Efficient Without Assumptions

Let us begin with some background.

## 4.1 The Problem

We are given a statistical ZK interactive proof system $(P, V)$ for $L$. We wish to construct a new statistical ZK interactive proof system $(\overline{P}, \overline{V})$ for $L$ whose prover is efficient. Let us begin by reviewing previous solutions to the problem and see whence arise the complexity assumptions that we wish to eliminate.

The solution of [5] begins with the observation that it is easy to construct an efficient prover $P_h$ such that the system $(P_h, V)$ is statistical ZK with respect to the honest verifier $V$ (that is, there exists a $P_h$-simulator for $V$). In our present terminology, this prover $P_h$ is simply the simulation based prover $P_{S_V}$ (more precisely, $P_h(x, \alpha_1\beta_1 \dots \alpha_t) = P_{S_V}(x, \lambda, \alpha_1\beta_1 \dots \alpha_t)$). The problem is that $(P_h, V)$ is not statistical ZK with respect to all verifiers as the definition of statistical ZK requires (that is, there does not necessarily exist a $P_h$-simulator for $\widehat{V} \neq V$). Indeed, sampling the output of $S_V$ in order to reply to a verifier $\widehat{V}$ does not yield a statistical ZK proof with respect to $\widehat{V}$.

What if, when talking to $\widehat{V}$, the prover should sample from the outputs of the simulator $S_{\widehat{V}}$ for this particular cheating verifier $\widehat{V}$? That is, when talking to $\widehat{V}$, the prover would be running $P_{S_{\widehat{V}}}$. This would yield the right distribution, but cannot be (efficiently) achieved. For we want to construct a single efficient prover. How could this prover know which (of the infinitely many) provers $P_{S_{\widehat{V}}}$ to run at any point in time? He cannot tell who he is talking to, and, anyway, cannot keep an infinite number of pieces of code handy.

In [5] the problem of the cheating verifier is surmounted by constructing a "compiler" which, given a proof system for $L$ which is statistical ZK with respect to the honest verifier, transforms it into a new proof system which is statistical ZK with respect to *any* (cheating) verifier, while preserving the power of the prover. They then apply this compiler to transform $(P_h, V)$. It is the compiler that uses a complexity assumption.

To summarize, their solution is in two steps: given a statistical ZK proof system $(P, V)$ for $L$,

*Step 1:* Construct an efficient prover $P_h$ such that $(P_h, V)$ is a proof system for $L$ which is statistical ZK with respect to the honest verifier $V$

*Step 2:* Using a complexity assumption, "compile" the $(P_h, V)$ proof into a proof $(\overline{P}, \overline{V})$ in which $\overline{P}$ is of the same complexity as $P_h$ but now $(\overline{P}, \overline{V})$ is statistical ZK with respect to *all* verifiers (as required by the definition of ZK).

All subsequent work on the question (until now) has followed the same two step paradigm, and (because of Step 2) has thereby used complexity assumptions. Below we will see that the problem of the cheating verifier can be viewed differently in this context, and the two step paradigm can be avoided. The complexity assumption is then unnecessary.

### 4.2 Our Solution

Our goal is to construct a PPT oracle machine $P_{\mathit{eff}}$ such that $(P_{\mathit{eff}}^{NP}, V)$ is a statistical zero-knowledge interactive proof system for $L$.

The first step is to define a single *universal* verifier who, in essence, can "simulate" the behavior of all verifiers as far as the prover is concerned. The key in the definition of the universal verifier $V^*$ (as in several proofs in [15, 23]) is to make appropriate use of the auxiliary inputs.

**Definition 4.1** *The* universal verifier $V^*$ *(of $g$ rounds and message length $l$) is defined as follows. Let $x \in \{0,1\}^n$ and let $\alpha_1, \beta_1, \ldots, \alpha_{t-1}, \beta_{t-1}, \alpha_1^*, \ldots, \alpha_i^*$ be $l(n)$ bit strings, $t, i \leq g$. On input $x$ (common input), $\alpha_1^* \ldots \alpha_i^*$ (auxiliary input), $\alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}$ (conversation so far), and $R$ (random tape) the message $V^*(x, \alpha_1^* \ldots \alpha_i^*, \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}, R)$ that $V^*$ computes (and sends to the prover) is*

$$\begin{cases} \alpha_t^* & \text{if } 1 \leq t \leq i \\ 0^l & \text{otherwise}. \end{cases}$$

*At the end of the interaction, $V^*$ accepts.*

In other words, the universal verifier's response in the $t$-th round is the $t$-th string written on his auxiliary input as long as there are $\geq t$ strings on his auxiliary input, and $0^l$ otherwise. Note that this response of $V^*$ is independent of the conversation $\alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}$ so far, and that $V^*$ is deterministic (random tape length 0). Also note that at the end of the protocol $V^*$ blindly accepts.

Since $(P, V)$ is ZK we know that there exists a ZK $P$-simulator $S^*$ for $V^*$. Note that outputs of $S^*$ have the form $(\lambda, \alpha_1\beta_1 \ldots \alpha_g\beta_g)$ where $\lambda$ is the empty string.

**Definition 4.2** *Let $V^*$ be the universal verifier and $S^*$ a (statistical) ZK $P$-simulator for $V^*$. We define $P^*$ by*

$$P^*(x, \alpha_1\beta_1 \ldots \alpha_t) \;=\; P_{S^*}(x, \alpha_1 \ldots \alpha_t, \alpha_1\beta_1 \ldots \alpha_t) \;,$$

*where $P_{S^*}$ is the simulation based prover for $S^*$. We call $P^*$ a* universal *prover for $(P, V)$.*

It is to be understood that if $(P, V)$ is perfect ZK then $S^*$ is a perfect ZK $P$-simulator. Let us now try to give some intuition for these definitions and their use.

Fix a common input $x$, and suppose we want to compute $P(x, \alpha_1\beta_1 \ldots \alpha_t)$, the prover's response when the history is $\alpha_1\beta_1 \ldots \alpha_t$. The idea is that since this response $P(x, \alpha_1\beta_1 \ldots \alpha_t)$ depends *only* on the history $\alpha_1\beta_1 \ldots \alpha_t$, it does not matter how the verifier is computing his messages as long as the history is indeed $\alpha_1\beta_1 \ldots \alpha_t$. In particular, we may as well assume $P$ is talking to the universal verifier with the latter's auxiliary input set to $\alpha_1\alpha_2 \ldots \alpha_t$. But the result of $P$ talking to $V^*$ (on that auxiliary input) is mimicked by $S^*(x, \alpha_1\alpha_2 \ldots \alpha_t)$, and $P_{S^*}$ just samples this last distribution to appropriately extract the prover's next message. So we can compute $P$ using $P^*$. In fact, if $(P, V)$ were perfect (rather than statistical) ZK then $P^*$ would equal $P$ (recall that both are probabilistic functions, so that this equality means that on any input the distribution given by both functions is identical). That is, the new simulation based prover is just another way of looking at the original prover $P$. When $(P, V)$ is statistical (rather than perfect) ZK, $P^*$ and $P$ are sufficiently close. The following Lemma states this more precisely.

**Lemma 4.3** *Let $x \in L \cap \{0, 1\}^n$ and let $\alpha_1\beta_1 \ldots \alpha_t$ be $l(n)$ bit strings (the conversation so far).*

(1) *If $(P, V)$ is statistical ZK then the distributions $P^*(x, \alpha_1\beta_1 \ldots \alpha_t)$ and $P(x, \alpha_1\beta_1 \ldots \alpha_t)$ are $\delta$-close, where $\delta$ is the distance between $S^*(x, \alpha_1\alpha_2 \ldots \alpha_t)$ and* $\mathrm{View}_{(P,V^*)}(x, \alpha_1\alpha_2 \ldots \alpha_t)$.

(2) *If $(P, V)$ is perfect ZK then $P^*(x, \alpha_1\beta_1 \ldots \alpha_t) = P(x, \alpha_1\beta_1 \ldots \alpha_t)$.*

**Proof:** We prove (2), and omit the simple extension of this argument required to establish (1). Consider the distribution $D(x, \alpha_1\beta_1 \ldots \alpha_t)$ on $\{0, 1\}^l$ given by the following experiment:

(1) Pick a random element $(\lambda, \alpha_1'\beta_1' \ldots \alpha_g'\beta_g')$ of $\mathit{View}_{(P,V^*)}(x, \alpha_1\alpha_2 \ldots \alpha_t)$ subject to the constraint that $\alpha_1'\beta_1' \ldots \alpha_t' = \alpha_1\beta_1 \ldots \alpha_t$

(2) Output $\beta_t'$.

The definition of the universal verifier implies that $D(x, \alpha_1\beta_1 \ldots \alpha_t) = P(x, \alpha_1\beta_1 \ldots \alpha_t)$. But $S^*(x, \alpha_1\alpha_2 \ldots \alpha_t) = \mathit{View}_{(P,V^*)}(x, \alpha_1\alpha_2 \ldots \alpha_t)$. By the definition of $P_{S^*}$ (cf. Definition 3.2) it follows that $P_{S^*}(x, \alpha_1 \ldots \alpha_t, \alpha_1\beta_1 \ldots \alpha_t) = P(x, \alpha_1\beta_1 \ldots \alpha_t)$. But $P^*(x, \alpha_1\beta_1 \ldots \alpha_t) = P_{S^*}(x, \alpha_1 \ldots \alpha_t, \alpha_1\beta_1 \ldots \alpha_t)$. ■

Note that $\delta$ is negligible.

Combining this Lemma with Theorem 3.3 yields the desired conclusion.

**Theorem 4.4** *Suppose $L$ has a statistical ZK interactive proof sytem $(P, V)$. Then there is PPT oracle machine $P_{\mathit{eff}}$ such that $(P_{\mathit{eff}}^{NP}, V)$ is a statistical ZK interactive proof system for $L$.*

**Proof:** Let $d^*(n)$ equal

$$\max_{x \in L, |x|=n} \sup_{a \in \{0,1\}^*} d(S^*(x, a), \mathit{View}_{(P,V^*)}(x, a)) \;.$$

We know that this function is negligible.

Let $P^*$ be a universal prover for $(P, V)$. Let $\widehat{V}$ be (any) verifier. Let $x \in L \cap \{0, 1\}^n$ and $a \in \{0, 1\}^*$. Lemma 4.3 (1) implies that the distance between $\mathit{View}_{(P,\widehat{V})}(x, a)$ and $\mathit{View}_{(P^*,\widehat{V})}(x, a)$ is at most $g(n)d^*(n)$. Note that since $d^*$ is negligible, so is $gd^*$.

This implies that $(P^*, V)$ is a statistical ZK interactive proof system for $L$. To see this first note that setting $\widehat{V}$ in the above statement to be the honest verifier $V$ implies that the completeness condition holds for $(P^*, V)$. Since $V$ is unchanged the soundness of course still holds. And if $S_{\widehat{V}}$ is a statistical ZK $P$-simulator for $\widehat{V}$ then it is also a statistical ZK $P^*$-simulator for $\widehat{V}$.
Finally we note that by Theorem 3.3 a distribution within negligible distance of $P^*(x, \alpha_1\beta_1 \ldots \alpha_t) = P_{S^*}(x, \alpha_1 \ldots \alpha_t, \alpha_1\beta_1 \ldots \alpha_t)$ is computable in PPT with an NP oracle, and one can check that the above argument still applies with the addition of some negligible distances. ■

We remark that perfect ZK can be preserved at the computational price of allowing $P_{\mathit{eff}}$ to have a $\Sigma_2^P$ oracle instead of an NP one:

**Theorem 4.5** *Suppose $L$ has a perfect ZK interactive proof sytem $(P, V)$. Then there is PPT oracle machine $P_{\mathit{eff}}$ such that $(P_{\mathit{eff}}^{\Sigma_2^{\mathrm{P}}}, V)$ is a perfect ZK interactive proof system for $L$.*

**Proof:** Use the same argument as in the proof of the previous theorem (except of course use Lemma 4.3 (2) rather than Lemma 4.3 (1)), but implement $P_{S^*}$ using the uniform generation algorithm of [21] (rather than the almost uniform generation of Theorem A.2). ■

When we allow simulators to run in expected (rather than fixed) PPT, the statements of our theorems change to allow the oracle machine $P_{\mathit{eff}}$ too to run in expected (rather than fixed) PPT.

It is possible to extend these techniques to derive bounds on the complexity of the prover in a proof of knowledge complexity $k(n) > 0$ (cf. Definition 2.8) while preserving the KC. However, the bounds we get are not nearly as good: a straightforward application of our techniques shows only that if $L$ has an interactive proof of knowledge complexity $k(n) \leq n^{O(1)}$ then $L$ has an interactive proof $(P, V)$ of knowledge complexity $k(n)$ in which $P$ is a probabilistic, exponential time machine with a NEXP oracle. Not only is the complexity much greater than for ZK, but we don't know how to do better even if we assume $k(n)$ is just 1 (rather than zero). Deriving better bounds on the complexity of non-zero KC provers while preserving the KC is another open question.

In the next section we will see however that if we don't want to preserve the KC then better bounds *are* achievable, and this has applications to bounding the complexity of low KC languages.

# 5 Bounds on the Complexity of Low KC languages

In this section we present a result which ties the knowledge complexity of an interactive proof for a language $L$ to the time complexity of $L$. This result relies on an extension to positive KC of our techniques from the previous sections.

Throughout this paper, we use the definition of knowledge complexity in the *fraction* sense [16]. Loosely speaking, we say that a protocol $(P, V)$ has knowledge complexity $k(n)$ if for any verifier $\widehat{V}$ there exists a simulator $S_{\widehat{V}}$ with the following property. For any $x \in L$ and auxiliary input $a \in \{0,1\}^*$ there exists a subspace (denoted $SUCC_{x,a}$) of the set $\{0,1\}^{p(n)}$ of the simulator's possible random tapes which has density $\geq 2^{-k(n)}$ (i.e. if we choose a tape at random from $\{0,1\}^{p(n)}$ then this tape is in $SUCC_{x,a}$ with probability $\geq 2^{-k(n)}$), and the output of the simulator on input $x, a$ and a uniformly chosen random tape in $SUCC_{x,a}$ is statistically close to a random element of $View_{(P,\widehat{V})}(x, a)$. The formal definition appears in §2.

There are two other definitions of non-zero KC: the oracle definition and the hint definition [16]. It is shown in [16] that the oracle definition is equal up to an additive constant to the fraction definition, and therefore our results apply to the oracle definition as well.

Our results apply also to the hint definition. However, with this definition, it is easy to prove even stronger results, and such stronger results already appear in [16]. Nevertheless, several reasons are presented in [16] why not to regard the hint definition as an appropriate measure.

Recall (Definition 2.3) that we have adopted the convention that an interactive proof has a negligible error probability. Let us now proceed to our result.

## 5.1 Overview

We are given a $g(n)$ round interactive proof system $(P, V)$ for $L$ with statistical KC $k(n)$, and we suppose $g(n)k(n) = O(\log n)$. Our goal is to show that $L \in \mathrm{BPP}^{NP}$. We begin with an overview of the proof.

We prove more than required. We show that under the given conditions, there exists an interactive proof for $L$ in which the prover is a PPT machine with an access to an NP oracle. As the prover has the power to run the verifier as well, this implies that $L \in \mathrm{BPP}^{\mathrm{NP}}$.

Let $S$ be a statistical KC $k(n)$ $P$-simulator for the honest verifier $V$ (cf. Definition 2.7), and $P_S$ the simulation based prover for $S$ (cf. Definition 3.2). We let $P^*$ be the prover derived from $P_S$ by setting the auxiliary input to the empty string: $P^*(x, \alpha_1\beta_1 \ldots \alpha_t) = P_S(x, \lambda, \alpha_1\beta_1 \ldots \alpha_t)$ (intuitively, think of $P^*$ as being $P_S$; the difference is only a technicality). We show that $(P^*, V)$ has the following "separability" property: (1) if $x \in L$ then $V$, interacting with $P^*$, accepts with a probability that is greater than $n^{-O(1)}$, whereas, (2) if $x \notin L$ then $V$, interacting with any prover, accepts with a negligible probability. This is Lemma 5.2 (below).

Next, we use Theorem 3.3 to show that there is PPT machine with an NP oracle that can compute a distribution close enough to $P^*$ such that the separability property is maintained. This gives us a PPT with NP oracle prover $\overline{P}$ such that $(\overline{P}, V)$ has the separability property. Finally, we observe that standard amplification techniques can be used to transform $(\overline{P}, V)$ into an interactive proof system (cf. Definition 2.3) for $L$ without increasing the power of the prover.

**Remark 5.1** We note that although the definition of knowledge complexity (Definition 2.8) guarantees a KC $k(n)$ $P$-simulator for *each* possible verifier, we use only the KC $k(n)$ $P$-simulator for the honest verifier $V$. Therefore the validity of the theorem (Theorem 5.3) requires only that the interactive proof has knowledge complexity $k(n)$ with respect to the honest verifier.

## 5.2 Our Results

The main lemma is the following.

**Lemma 5.2** *Suppose $(P, V)$ is a $g(n)$ round interactive proof for $L$, and $S$ is a statistical KC $k(n)$ $P$-simulator for $V$. Suppose also that $g(n)k(n) = O(\log n)$. Define the prover $P^*$ by*

$$P^*(x, \alpha_1\beta_1 \ldots \alpha_t) \;=\; P_S(x, \lambda, \alpha_1\beta_1 \ldots \alpha_t)$$

*where $P_S$ is the simulation based prover for $S$. Then $(P^*, V)$ satisfies the following "separability" property:*

(1) *There is a constant $c \geq 0$ such that for any $x \in L$ the probability that $V$ accepts at the end of the interaction with $P^*$ on common input $x$ is $\geq |x|^{-c}$*

(2) *For any (cheating) prover $\widehat{P}$ and common input $x \notin L$ the probability that $V$ accepts at the end of the interaction with $\widehat{P}$ is negligible*

Note that we are making no claims about the knowledge complexity of the system $(P^*, V)$ constructed in the above lemma. This is in contrast to our results in §4 where we were trying to preserve the knowledge complexity (in the particular case where this knowledge complexity was zero).

The (technical) proof of Lemma 5.2 appears in Appendix B. Here let us give some intuition.

Condition (2) is immediate; the concern is condition (1). Fix an $x \in L$. Recall that when the random tape of the simulator $S$ is uniformly picked in $SUCC_{x,\lambda}$, the output distribution of $S$ is statistically close to the distribution of the conversations between $P$ and $V$. We show that our new prover $P^*$ (using $P_S$) picks its answer using a random tape in $SUCC_{x,\lambda}$ with probability $\geq 2^{-k}$. Therefore with probability at least $2^{-kg} = n^{-O(1)}$, $P_S$ picks all its answers using random tapes ¿from $SUCC_{x,\lambda}$. Intuitively, he thus gains an advantage which is $2^{-kg} = n^{-O(1)}$ times the advantage of the original prover $P$. However this is a simplification, because we must deal with the fact that in all but the first round, $P_S$ is not sampling uniformly from the space of all coin tosses. Rather, he is sampling conditional to the prefix of the conversation so far being some particular value. A complete argument is in Appendix B.

Our theorem follows.

**Theorem 5.3** *Suppose $L$ has a $g(n)$ round interactive proof with statistical knowledge complexity $k(n)$, and suppose also that $g(n)k(n) = O(\log n)$. Then $L$ is in* $\text{BPP}^{\text{NP}}$.

**Proof:** By assumption there is a $g(n)$ round interactive proof $(P, V)$ with statistical knowledge complexity $k(n)$. It suffices to show that there is a PPT machine $P_{\mathit{eff}}$ such that $(P_{\mathit{eff}}^{NP}, V)$ is an interactive proof system for $L$.

Let $S$ be a statistical KC $k(n)$ $P$-simulator for $V$. Let $P_S$ be the corresponding simulation based prover, and let $P^*$ be as in Lemma 5.2. Then $(P^*, V)$ has the separability property. We let $\overline{P}$ denote the PPT with NP oracle prover which is given by using $T$ rather than $P_S$ in the definition of $P^*$, where $T$ is the machine that is guaranteed in Theorem 3.3, setting $\delta = 2^{-n}$ in that theorem so that the output of $T$ is $2^{-n}$-close to the output of $P_S$.

Let $x \in L$. Observe that $View_{(P^*,V)}(x, \lambda)$ and $View_{(\overline{P},V)}(x, \lambda)$ are $g2^{-n}$-close, and thus, on common input $x$, the difference between the probability that $P^*$ convinces $V$ to accept and the probability that $\overline{P}$ convinces $V$ to accept is at most $g2^{-n}$. Since $g2^{-n}$ is negligible, this means that the separability property still holds:

(1) There is a constant $c \geq 0$ such that for any $x \in L$ the probability that $V$ accepts at the end of the interaction with $\overline{P}$ on common input $x$ is $\geq |x|^{-c}$

(2) For any (cheating) prover $\widehat{P}$ and common input $x \notin L$ the probability that $V$ accepts at the end of the interaction with $\widehat{P}$ is negligible.

Finally, standard amplification techniques can be used to turn $(\overline{P}, V)$ into an interactive proof for $L$ without increasing the complexity of the prover. ■

# 6 Conclusions and Open Problems

We showed that if $L$ has a statistical ZK proof then it has a statistical ZK proof with a prover who runs in PPT with an NP oracle. This was previously only known given complexity assumptions. Our first question is whether one can remove assumptions from other similar problems. In particular, can one unconditionally establish any of the following?

(1) If $L$ has a statistical ZK proof then it has a statistical ZK proof with perfect completeness (i.e. the verifier accepts with probability 1 when $x \in L$; cf. [14])

(2) If $L$ has a statistical ZK proof then it has a statistical ZK proof with a black-box simulator.

(3) If $L$ has an interactive proof which is statistical ZK with respect to the honest verifier, then it has a statistical ZK interactive proof.

We recall that these results are known given complexity assumptions [5].

Second, on the subject of the power of the prover. Our bound of PPT with NP oracle does not depend on the complexity of the language. Can one find "tighter" relationships between the complexity of $L$ and the complexity of a statistical ZK prover for $L$? For example, what can one say about statistical ZK in the model of [4] where the prover is PPT with an oracle for $L$?

We also showed that if a language $L$ has a $g(n)$ round interactive proof with statistical knowledge complexity $k(n)$, and if $g(n)k(n) = O(\log n)$ then $L \in \mathrm{BPP}^{NP}$. Is the restriction on rounds necessary? What is the complexity of languages of KC $O(\log n)$? How about just KC 1?

## Acknowledgments

We thank Oded Goldreich and Avi Wigderson for helpful remarks and proof simplifications.

## References

[1] W. Aiello and J. Håstad. Perfect Zero-Knowledge can be Recognized in Two Rounds. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[2] L. Babai. Trading Group Theory for Randomness. *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing,* ACM (1985).

[3] L. Babai, L. Fortnow and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

[4] R. Beigel, M. Bellare, J. Feigenbaum and S. Goldwasser. Languages that are Easier than their Proofs. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991).

[5] M. Bellare, S. Micali and R. Ostrovsky. The (True) Complexity of Statistical Zero-Knowledge. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[6] M. Bellare and E. Petrank. Making Zero-Knowledge Provers Efficient. Technical Report, Computer Science Department, Technion, Haifa, Israel.

[7] M. Ben-Or, S. Goldwasser, O. Goldreich, J. Håstad, J. Kilian, S. Micali and P. Rogaway. Everything Provable is Provable in Zero-Knowledge. *Advances in Cryptology — Proceedings of CRYPTO 88,* Lecture Notes in Computer Science 403, Springer-Verlag (1989). S. Goldwasser, ed.

[8] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing,* ACM (1988).

[9] M. Blum and S. Kannan. Designing Programs that Check their Work. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[10] L. Carter and M. Wegman. Universal Classes of Hash Functions. *J. Computer and System Sciences* **18**, 143–154 (1979).

[11] U. Feige. Interactive Proofs. M.Sc Thesis, Weizmann Institute of Science. August 1987.

[12] L. Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research (ed. S. Micali)* Vol. 18 (1989).

[13] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *Proceedings of ICALP 90*.

[14] O. Goldreich, Y. Mansour and M. Sipser. Interactive Proof Systems: Provers that never Fail and Random Selection. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[15] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. Technical Report #570, Technion (1989).

[16] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991).

[17] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing,* ACM (1985).

[18] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *SIAM J. Comput.* **18** (1), 186-208 (February 1989).

[19] R. Impagliazzo, L. Levin and M. Luby. Pseudo-Random Generation from One-Way Functions. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[20] R. Impagliazzo and M. Yung. Direct Minimum-Knowledge computations. *Advances in Cryptology — Proceedings of CRYPTO 87,* Lecture Notes in Computer Science 293, Springer-Verlag (1987).

[21] M. Jerrum, L. Valiant and V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science* **43**, 169-188 (1986).

[22] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

[23] Y. Oren. On The Cunning Power of Cheating Verifiers: Some Observations About Zero Knowledge Proofs. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[24] R. Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. Structures 1991.

[25] R. Ostrovsky, R. Venkatesan and M. Yung. On the Complexity of Asymmetric Games. Manuscript (1990).

[26] M. Sipser. A Complexity Theoretic Approach to Randomness. *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing,* ACM (1983).

[27] L. Stockmeyer. The Complexity of Approximate Counting. *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing,* ACM (1983).

[28] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Proofs of Possession of Information. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

## A Efficient Almost Uniform Generation

Let $M$ be a TM. We let $M_n \stackrel{\text{def}}{=} \{r \in \{0,1\}^n : M(r) = 1\}$, and we let $\langle M \rangle$ denote an encoding of $M$. We also let $\mathcal{U}[B]$ denote the uniform distribution over a finite set $B$.

**Definition A.1** *A* universal almost uniform generator *is a (probabilistic) machine $A$ which has the property that $A(1^n, \delta, \langle M \rangle)$ and $\mathcal{U}[M_n]$ are $\delta$-close, for all $n \in \mathsf{N}, \delta > 0$ and TMs $M$.*

Our result is the following.

**Theorem A.2** *There is a probabilistic, oracle machine $U$ such $U^{NP}$ is a universal almost uniform generator, and moreover, the running time of $U^{NP}$ on inputs $1^n, \delta, \langle M \rangle$ is polynomial in $n, \lg \delta^{-1}$ and $T_M(n)$.*

Here $T_M(n)$ denotes the maximum, over all $r \in \{0,1\}^n$, of the running time of $M$ on input $r$.

Note that the running time of the procedure of Theorem A.2 is polynomial in $\lg \delta^{-1}$ rather than $\delta^{-1}$, so that in time polynomial in $n$ we can achieve exponentially small (in $n$) error.

The proof of Theorem A.2 is derived by combining techniques from [21, 27, 26]. We use Jerrum, Valiant and Vazirani's PPT reduction of uniform generation to approximate counting [21]. The key is the strength of the [21] reduction: it is capable of achieving distributions within an exponentially small distance of uniform given a primitive for estimating set sizes (namely approximate counting) that is only accurate to within the reciprocal of a polynomial. Whereas approximate counting took polynomial time with a $\Sigma_2^P$ oracle (Stockmeyer [27]) we show that its "almost" version can be implemented in PPT with an NP oracle using hashing techniques [26]. We then note that the reduction of [21] still goes through. For completeness we provide the details below.

We use universal hash functions [10]. Let $H_{k,b}$ denote the set of all maps $h_{A,\alpha}\{0,1\}^k \to \{0,1\}^b$ given by $h_{A,\alpha}(y) = Ay + \alpha$ where $A$ is a $b$ by $k$ matrix over GF[2] and $\alpha$ is a $b$-vector over GF[2], and the arithmetic is over GF[2]. The following lemma follows ¿from Lemma 4.1 in Aiello-Håstad [1], which in turn is based on ideas of Babai [2] and Sipser [26].

**Lemma A.3** *Let $b, k$ be positive integers, $\delta > 0$ and $S \subseteq \{0,1\}^k$. Let $l = 16 \lg(2\delta^{-1})$. Select random and independent hash functions $h_1, \ldots, h_{2l} \in H_{k,b}$ and random and independent strings $z_1, \ldots, z_{2l} \in \{0,1\}^b$. Let the random variable $X$ equal the number of indices $i$ for which $h_i^{-1}(z_i) \cap S \neq \emptyset$. Then the probability that $X > l$ is*

(1) $\geq 1 - \delta$ *if* $|S| \geq 4 \cdot 2^b$

(2) $\leq \delta$ *if* $|S| \leq 2^b/4$.

The notion of approximate counting we are interested in is as follows (cf. [27]).

**Definition A.4** *A* universal approximate counter *is a (probabilistic) machine $C$ which, on inputs $1^n, \epsilon > 0, \delta > 0, \langle M \rangle$ outputs an estimate $v$ which, with probability $\geq 1 - \delta$, satisfies $|M_n|/(1+\epsilon) \leq v \leq |M_n|(1+\epsilon)$.*

Let us now establish the result on approximate counting that we need.

**Theorem A.5** *There is a probabilistic, oracle machine $C$ such $C^{NP}$ is a universal approximate counter, and moreover, the running time of $C^{NP}$ on inputs $1^n, \epsilon, \delta, \langle M \rangle$ is polynomial in $n, \epsilon^{-1}, \lg \delta^{-1}$ and $T_M(n)$.*

**Proof:** The algorithm $C$ is given in Figure 1. The idea behind it is as follows.

$C^{NP}(1^n, \epsilon, \delta, \langle M \rangle)$
Choose $t$ such that $8^{1/t} \leq 1 + \epsilon$
{ **Comment** $t$ will be polynomial in $\epsilon^{-1}$ }
Let $k = nt$ and let $N = M^t$ be the machine defined by $N(y_1, \ldots, y_t) = 1$ iff $M(y_1) = \cdots = M(y_t) = 1$ and all the $y_i$ have the same length.
{ **Comment** $N_k = M_n^t$ }
$b \leftarrow 0$ ; $l \leftarrow 16 \lg(2k\delta^{-1})$
`repeat`
  $b \leftarrow b + 1$
  Pick at random $h_1, \ldots, h_{2l} \in H_{k,b}$ and $z_1, \ldots, z_{2l} \in \{0,1\}^b$. Use the NP oracle to determine, for each $i$, whether or not $h_i^{-1}(z_i) \cap N_k = \emptyset$, and let $X$ be the number of indices $i$ for which $h_i^{-1}(z_i) \cap N_k \neq \emptyset$.
`until` $(X \leq l$ or $b = k)$
{ **Comment** $\Pr[2^{b-1}/4 < |N_k| < 4 \cdot 2^b] \geq 1 - \delta$ }
Let $\alpha = 2^b$.
{ **Comment** $\Pr[|N_k|/4 < \alpha < 8|N_k|] \geq 1 - \delta$ }
Output $\alpha^{1/t}$
{ **Comment** $\Pr[|M_n|/(1+\epsilon) < \alpha < |M_n|(1+\epsilon)] \geq 1 - \delta$ }

Figure 1: Universal Approximate Counting in PPT with an NP Oracle

We first claim that given $\langle N \rangle$ and our NP oracle we can find, in time polynomial in $k, \lg \delta^{-1}$ and $T_N(k)$, an estimate which with probability $\geq 1 - \delta$ lies between $|N_k|/4$ and $8|N_k|$. The ability to do this much weaker approximation suffices. For we can let $N = M^t$ and use the weak approximation to obtain an estimate $\alpha$ satisfying $|M_n^t|/4 \leq \alpha \leq 8|M_n^t|$, whence $4^{-1/t}|M_n| \leq \alpha^{1/t} \leq 8^{1/t}|M_n|$, and by choosing $t$ so that $8^{1/t} \leq 1 + \epsilon$ ($t$ polynomial in $\epsilon^{-1}$ suffices) we can output $\alpha^{1/t}$ as the estimate. So it remains to justify the claim.

Let $l = 16 \lg(k\delta^{-1})$. Suppose $b$ between 1 and $k$ is fixed and let $S \stackrel{\text{def}}{=} N_k \subseteq \{0,1\}^k$. If we compute the random variable $X$ according to the experiment specified in Lemma A.3 (note that this can be done in time polynomial in $k, \lg \delta^{-1}$ and $T_N(k)$ given access to an NP oracle) then the probability that $X > l$ is

(1) $\geq 1 - \delta/k$ if $|N_k| \geq 4 \cdot 2^b$
(2) $\leq \delta/k$ if $|N_k| \leq 2^b/4$.

The loop in the algorithm of Figure 1 finds the first $b$ between 1 and $k$ such that $X \leq l$ if one exists, and otherwise lets $b = k$. In either case we have $\Pr[2^{b-1}/4 < |N_k| < 4 \cdot 2^b] \geq 1 - \delta$. Setting $\alpha = 2^b$ we get $\Pr[|N_k|/4 \leq \alpha \leq 8|N_k|] \geq 1 - \delta$, as desired. ■

Notice that the estimates from the above theorem are only accurate to within the reciprocal of a polynomial (that is, $\epsilon^{-1}$ must be bounded by a polynomial in $n$). It is a feature of the [21] reduction that this suffices. To complete the proof of Theorem A.2 based on the above theorem, let us sketch the use of this reduction. Briefly, this works as follows. Given $1^n, \delta, \langle M \rangle$ we construct a random element of $M_n$ bit by bit. If $\alpha$ is the current prefix, we estimate the number of elements of $M_n$ with prefix $\alpha b$ for each $b \in \{0,1\}$, and use these estimates to extend $\alpha$. We replace the approximate counting used for these estimates with the one guaranteed by Theorem A.5, setting $\epsilon$ to be as in [21] and setting $\delta$ (for the counting) to be our given $\delta$ divided by $2n$. This guarantees that all estimates are within the desired bound with probability $\geq 1 - \delta$. The distance between the distributions is at most $\delta$ as desired.

# B Proof of Lemma 5.2

Let us restate Lemma 5.2 and this time provide the proof.

**Lemma 5.2** *Suppose $(P, V)$ is a $g(n)$ round interactive proof for $L$, and $S$ is a statistical KC $k(n)$ $P$-simulator for $V$. Suppose also that $g(n)k(n) = O(\log n)$. Define the prover $P^*$ by*

$$P^*(x, \alpha_1\beta_1 \dots \alpha_t) \;=\; P_S(x, \lambda, \alpha_1\beta_1 \dots \alpha_t)$$

*where $P_S$ is the simulation based prover for $S$. Then $(P^*, V)$ satisfies the following "separability" property:*

(1) *There is a constant $c \geq 0$ such that for any $x \in L$ the probability that $V$ accepts at the end of the interaction with $P^*$ on common input $x$ is $\geq |x|^{-c}$*

(2) *For any (cheating) prover $\widehat{P}$ and common input $x \notin L$ the probability that $V$ accepts at the end of the interaction with $\widehat{P}$ is negligible*

**Proof:** Condition (2) follows directly from the soundness of $(P, V)$. We concentrate on (1). For simplicitly, we first present the proof under the assumption that $S$ is a perfect (rather than just statistical) KC $k(n)$ $P$-simulator for $V$. The proof of the more general case follows thereafter.

Let $p$ be the number of coin tosses of $S$. We call a string $c$ a $t$-th round prover prefix if $c = \alpha_1\beta_1 \dots \alpha_t$ for some $l$ bit strings $\alpha_1, \beta_1, \dots, \alpha_t$. We let $R(c) \stackrel{\text{def}}{=} \{\, r \in \{0,1\}^{p(n)} : S_t(x, \lambda, r) = c \,\}$ denote the set of random tapes of the simulator that are consistent with a $t$-th round prover prefix $c$ .

We now analyze the interaction between $P^*$ and $V$ on input $x \in L$. Let us denote by $c_t$ the $t$-th round prover prefix of this interaction. Recall that $P_S(x, \lambda, c_t)$ picks $r$ at random from $R(c_t)$, computes $S(x, \lambda, r)$, and outputs $\beta'_t$, where $S(x, \lambda, r) = (R, \alpha_1\beta_1 \dots \alpha_t\beta'_t \dots \alpha'_g\beta'_g)$ (cf. Definition 3.2). Let us denote by $r_t$ the string $r$ that was chosen by $P_S$ in the $t-$th round. For $t = 1, \dots, g$ let us denote by $\mathrm{GOOD_t}$ the event that $r_j \in \mathit{SUCC}_{x,\lambda}$ for all $j = 1, \dots, t$. Fix $\mathrm{GOOD_0}$ to be some event with probability 1.

We claim that for each round $t = 1, \dots, g$ it is the case that

$$\Pr[\mathrm{GOOD_t} | \mathrm{GOOD_{t-1}}] \geq 2^{-k} \; . \tag{1}$$

It follows that $\Pr[\mathrm{GOOD_g}] \geq 2^{-kg} \geq n^{-d}$, where $d$ is a constant such that $g(n)k(n) \leq d \lg n$. Since $S$ is, by assumption, a perfect KC $k(n)$ $P$-simulator for $V$ we know that it outputs the distribution $\mathit{View}_{(P,V)}(x, \lambda)$ when its random tape $r$ is chosen uniformly from $\mathit{SUCC}_{x,\lambda}$ (cf. Definition 2.7). As $P_S$ chooses each (possible) random tape with equal probability, the event $\mathrm{GOOD_g}$ being true implies that $r_t$ is uniformly distributed in $\mathit{SUCC}_{x,\lambda} \cap R(c_t)$ for all rounds $t$. Therefore, the subspace of $\mathit{View}_{(P^*,V)}(x, \lambda)$ obtained by conditioning on the event $\mathrm{GOOD_g}$ equals $\mathit{View}_{(P,V)}(x, \lambda)$. So the probability that $V$ accepts in the interaction with $P^*$ is at least $\Pr[\mathrm{GOOD_g}]$ times the probability that $V$ accepts in the interaction with $P$. So the above together with the completeness of $(P, V)$ implies that the probability that $V$ accepts in the interaction with $P^*$ is $\geq n^{-O(1)}$, as desired.

It remains to justify Equation (1). It is by definition true that if we sample the space of random tapes of the simulator once, we hit $\mathit{SUCC}_{x,\lambda}$ with probability at least $2^{-k}$. However, when we are sampling (uniformly) a subset of the possible random tapes of the simulator (the subset that is consistent with the conversation so far), it is not necessarily true that we hit $\mathit{SUCC}_{x,\lambda}$ with probability $\geq 2^{-k}$. Still, Equation (1) can be derived by analyzing the prefix of the conversation (and thus the subset of the simulators possible coin tosses) as a stochastic variable.

Fix a round $t$ and assume $\mathrm{GOOD_{t-1}}$. We will now bound the probability of $\overline{\mathrm{GOOD}}_t$ (the complement of $\mathrm{GOOD_t}$).

We call a string $c$ a $t$-th round verifier prefix if $c = \alpha_1\beta_1 \dots \alpha_{t-1}\beta_{t-1}$ for some $l$ bit strings

$\alpha_1, \beta_1, \ldots, \beta_{t-1}$. Let $M = 2^{2(t-1)l}$ and let $w_1, \ldots, w_M$ be an enumeration of the set of all possible $t$-th round verifier prefixes. Let $B$ be the bad set of the simulator's coins (i.e. the complement of $SUCC_{x,\lambda}$). Let $B_i$ denote the coin tosses from $B$ that are consistent with $w_i$ (i.e. running the simulator on an element of $B_i$ produces a conversation whose $t$-th round verifier prefix is $w_i$) and similarly let $G_i$ denote the coin tosses from $SUCC_{x,\lambda}$ that are consistent with $w_i$.

Now we further partition these sets according to the values that could be taken on by the verifier's $t$-th message. Let $m = 2^l$ and let $a_1, \ldots, a_m$ be an enumeration of the set $\{0,1\}^l$ of all possible responses of the verifier in the $t$-th round. We let $B_i^j$ and $G_i^j$ denote the elements of $B_i$ and $G_i$, respectively, that are consistent with the verifier's next message being $a_j$, for $j = 1, \ldots, m$ and $i = 1, \ldots, M$. We let

$$\begin{aligned} b_i^j &= \Pr[B_i^j] \\ g_i^j &= \Pr[G_i^j] \\ b_i &= \Pr[B_i] \;=\; b_i^1 + \cdots + b_i^m \\ g_i &= \Pr[G_i] \;=\; g_i^1 + \cdots + g_i^m \end{aligned}$$

and we note that

$$\sum_{i=1}^{M} g_i \;=\; 2^{-k} \qquad \text{and} \qquad \sum_{i=1}^{M} b_i \;=\; 1 - 2^{-k}\ . \tag{2}$$

Our assumption that $\mathrm{GOOD}_{t-1}$ holds implies that $w_i$ appears with probability $2^k g_i$. Given this, the probability that $r_t \notin SUCC_{x,\lambda}$ is

$$\sum_{i=1}^{M} 2^k g_i \sum_{j=1}^{m} \frac{g_i^j}{g_i} \cdot \frac{b_i^j}{b_i^j + g_i^j}\ . \tag{3}$$

Here $b_i^j/(b_i^j + g_i^j)$ is the probability that $P_S$ samples ¿from the bad set in replying to $a_j$ when the conversation so far was $w_i$, and $g_i^j/g_i$ is the probability that the verifier sends $a_j$ given that the conversation so far was $w_i$.

We will now maximize the expression of Equation (3) subject to the constraints of Equation (2). First, we fix a value of $i$ and note that the Lagrange multiplier theorem implies that the maximum of

$$\sum_{j=1}^{m} \frac{b_i^j g_i^j}{b_i^j + g_i^j}$$

subject to the constraints $\sum_{j=1}^{m} g_i^j = g_i$ and $\sum_{j=1}^{m} b_i^j = b_i$ is $b_i g_i/(b_i + g_i)$. Substituting this in Equation (3) we are reduced to maximizing

$$\sum_{i=1}^{M} 2^k \frac{b_i g_i}{b_i + g_i}$$

subject to the constraints of Equation (2), and again using Lagrange multipliers we get that this is at most $1 - 2^{-k}$ as desired. ■

## B.1 The case of statistical knowledge complexity

The case of statistical knowledge complexity is a little more involved. Again (using the notaions of the above proof), we have to prove that $\mathrm{GOOD_g}$ has a large enough probability (Claim B.2 bellow), and that given $\mathrm{GOOD_g}$, $V$ accepts with high probability (Claim B.1 bellow).

As a mental experiment, we consider a prover and a verifier, that do not appear in the interaction, but their protocol will help us analyze the protocol between the simulation based prover and the original verifier. Recall that we denote by $P^*$ the simulation based prover (with a null auxiliary input to the simulator, cf. Definition 3.2 and Lemma 3.3). Denote by $V^*$ the analogue simualtion based verifier (in all references to the simulator in this section, we always grant it with a null auxiliary input). Note that the protocol $(P^*, V^*)$ is exactly equal to the output of the simulator. Let us further define $[P^*|SUCC_x]$ and $[V^*|SUCC_x]$ as the simulation based prover and verifier, given $SUCC_x$. That is, the random tape that is chosen for the simualtor in step (1) of definition Definition 3.2 is chosen (randomly and uniformly) from $SUCC_x$. Clearly, $([P^*|SUCC_x]\ , [V^*|SUCC_x]\ )$ is equal to the output of the simulator on the subspace $SUCC_x$ of its random tapes, and is therefore statistically close to $(P, V)$. Note that $[V^*|SUCC_x]$ is a fictious entity which is not necessarily a (probabilistic) polynomial time function.

Let us state and prove the two claims that validate Lemma 5.2 for the statistical case.

**Claim B.1** *Given* $\mathrm{GOOD_g}$*, the probability that* $V$ *accepts on input* $x \in L$ *while talking to* $P^*$*, is* $\geq 1 - \epsilon$ *for some negligible fraction* $\epsilon$*.*

**Proof:** Think of this probability as the accepting probability of the interaction $([P^*|SUCC_x]\ , V)$ on input $x \in L$. We know that $([P^*|SUCC_x]\ , [V^*|SUCC_x]\ )$ is statistically close to $(P, V)$ (we consider only the case $x \in L$), and therefore $[P^*|SUCC_x]$ convinces $[V^*|SUCC_x]$ that $x \in L$ with probability $1 - \epsilon_1$ for some negligible fraction $\epsilon_1$. The question we are interested in is what is the probability that $[P^*|SUCC_x]$ convince $V$. We claim that the protocol $([P^*|SUCC_x]\ , V)$ is statistically close to $(P, V)$ and thus we get that this probability is close to 1 up to a negligible fraction. This claim follows from the general fact that for any two protocols $(A, B)$ and $(A', B')$ that are statistically close, the protocol $(A, B')$ is statistically close to both. This fact can be proven by simple induction. ■

**Claim B.2**

$$\Pr[\mathrm{GOOD_g}] \geq n^{-O(1)}$$

**Proof:** We prove that for every $t$, $1 \leq t \leq g$,

$$\Pr[\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] \ \geq \ 2^{-k} - \epsilon_1$$

for some negligible fraction $\epsilon_1$. Thus we get that

$$\Pr[\mathrm{GOOD_g}] \ = \ \prod_{t=1}^{g} \Pr[\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] \ \geq \ 2^{-kg} - \epsilon$$

for some negligible fraction $\epsilon$, and we are done.

Since the probability that the simulator picks the next random string from $SUCC_x$ depends only on the conversation so far, we have

$$\Pr[\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] = \sum_{c_t} \Pr[\mathrm{GOOD_t}|c_t] \cdot \Pr[c_t|\mathrm{GOOD_{t-1}}] \quad (4)$$

where $c_t$ goes over all possible $t$-th round prover prefices. Intuitively, we show that the case of perfect knowledge complexity (which we have already proven) is "very close" to the case of statistical knowledge complexity.

Formally, we look again at the imaginary verifier $[V^*|SUCC_x]$ . We know that had the original protocol been $([P^*|SUCC_x]\ , [V^*|SUCC_x]\ )$ instead of $(P, V)$ then our simulator would have yielded a perfect simulation. Thus, in the protocol $(P^*, [V^*|SUCC_x]\ )$ we would have had $\Pr[\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] \geq$

$2^{-k}$. We have to analyze the protocol $(P^*, V)$, where $V$ has a behavior "close" to $[V^*|SUCC_x]$ . Let us look at round $t$ of the interaction, given that the simulator has used only random strings in $SUCC_x$ so far. By the same argument as in the previous claim, we have that, after $V$ has contributed the message $\alpha_t$ (as his message of round $t$), the prefices of $([P^*|SUCC_x]\ , [V^*|SUCC_x]\ )$ and $([P^*|SUCC_x]\ , V)$ are statistically close. The prover's behavior in this round, and in particular his picking a random string in $SUCC_x$ is identical is both protocols, given the conversation so far. Adding these facts to Equation (4) we get that

$$\sum_{c_t} \left| \Pr_{(P^*, V)} [\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] - \Pr_{(P^*, [V^*|SUCC_x]\ )} [\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] \right| < \epsilon_1$$

for some neglibile fraction $\epsilon_1$. Thus, we get that

$$\Pr_{(P^*, V)} [\mathrm{GOOD_t}|\mathrm{GOOD_{t-1}}] \geq 2^{-k} - \epsilon_1$$

and we are done. ■