An abridged version of this paper appears in *Proceedings of the Twenty Seventh Annual Symposium on the Theory of Computing,* ACM, 1995.

Knowledge on the Average— Perfect, Statistical and Logarithmic

William Aiello* Mihir Bellare† Ramarathnam Venkatesan*

March 22, 1995

Abstract

We initiate a broad investigation of the average measure of knowledge complexity suggested by Goldreich and Petrank [GP], and of a new and related measure of oracle entropy that we suggest here. Our main results are as follows.

We provide a new characterization of statistical zero-knowledge, as the class of languages leaking at most a negligible amount of *perfect* knowledge under the oracle entropy measure. We show that the oracle entropy of a language is always within an additive constant of its average knowledge complexity, thereby relating the new measure in a strong way to the older one. Finally, we show that the class of languages having proofs of logarithmic knowledge complexity, in the average or oracle entropy senses, is in $\mathsf{BPP}^{\mathsf{NP}}$.

In addition, we bound the amount of (perfect) knowledge leaked by perfect zero-knowledge proofs in which the simulator runs in expected polynomial time by $1/p(n)$ for an arbitrarily large polynomial $p(n)$. We also note some relations of oracle entropy to free bits in PCP. Finally, we provide protocols to prove highly accurate estimates of set sizes in constant rounds— these may be useful in other contexts.

* Math and Cryptography Research Group, Bellcore, 445 South St. Morristown, NJ 07960. E-mail: {aiello,venkie}@bellcore.com

† Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: mihir@cs.ucsd.edu

1 Introduction

The study of zero-knowledge [GMR] has proven to be fruitful for both complexity theory and cryptography. Quantifying knowledge above zero was suggested by [GMR]. Definitions for knowledge complexity (KC) emerged more recently, in the work of Goldreich and Petrank [GP].

Two measures of KC where defined in [GP]— a worst case measure and an average measure. Subsequent work has concentrated on the former. This paper launches a broad investigation of the latter. It also adds, and investigates, a third measure: an entropy based one called "oracle entropy." Our results indicate several compelling reasons to view the average and entropy measures as useful, interesting and important.

1.1 Discussion of main results

The perfect KC of statistical ZK. Our first set of results uses knowledge complexity to shed light on its parent notion of zero-knowledge. We provide a novel characterization of statistical zero-knowledge as the class of languages possessing proofs leaking only a *negligible* amount of *perfect* knowledge, in the oracle entropy sense. (Roughly, this means that a perfect simulation can be produced at the cost of revealing a negligible amount of information.) We also show that the amount of perfect knowledge revealed by a statistical ZK proof under the average oracle measure of [GP] is about one bit. These results capture a basic intuition about the notion of statistical zero-knowledge, namely that statistical is "almost perfect."

Oracle entropy. The new measure of oracle entropy is defined in the oracle framework of [GP] by appropriately blending Shannon type measures with simulation. Our main theorem concerning it shows that the new measure is within an additive constant of the more familiar average measure, which is important in indicating that we have not strayed too far from older notions. The new measure seems to be particularly relevant in discussing small quantities of knowledge above zero, where previous notions may be an overcount.

Proofs of logarithmic KC. We investigate the (time) complexity of languages of logarithmic knowledge complexity in the average or oracle entropy senses. We show that such languages are in $\mathsf{BPP}^{\mathsf{NP}}$. This extends [BP, GOP] who showed the same containment for the smaller class of languages of logarithmic KC in the strict, or worst case sense. We note that languages of logarithmic KC in the average or oracle entropy senses could have worst case KC which is polynomial, so that the extension is not direct— prior to our results the best one could say is that such languages were in PSPACE.

Other results. Other results concern strict versus expected poly-time simulation; connections of oracle entropy with free bits in PCP; and protocols for proving set sizes.

1.2 ZK and knowledge complexity

We now specify more precisely the relevant complexity classes so that in Section 1.3 we may provide precise theorem statements.

Zero-knowledge. We consider statistical and perfect (not computational) zero-knowledge. Denote by PZK (resp. SZK) the class of languages possessing perfect (resp. statistical) zero-knowledge interactive proofs of membership. We stress that these basic classes refer to the "full" definition in which the ZK is required to hold with respect to any (cheating) verifier— the corresponding classes where the ZK is required only to hold for the honest verifier are $\mathsf{SZK}_{\mathrm{hv}}$ and $\mathsf{PZK}_{\mathrm{hv}}$ respectively, and will also be discussed. Obviously $\mathsf{SZK} \subseteq \mathsf{SZK}_{\mathrm{hv}}$ and $\mathsf{PZK} \subseteq \mathsf{PZK}_{\mathrm{hv}}$.

KNOWLEDGE COMPLEXITY. We will consider three types, or measures, of knowledge complexity: the strict (or worst case) measure, the average measure, and the entropy based measure. (The first two are due to [GP] and the third is defined in this paper). Like zero-knowledge, each has its statistical and perfect versions. For $\kappa(n) \leq \text{poly}(n)$ and $\mathsf{type} \in \{\text{wc}, \text{av}, \text{en}\}$ we let $\mathsf{PKC}^{\mathsf{type}}[\kappa(n)]$ (resp. $\mathsf{SKC}^{\mathsf{type}}[\kappa(n)]$) denote the class of languages possessing interactive proofs of perfect (resp. statistical) knowledge complexity $\kappa(n)$ in the type sense. Again, in these classes the bound of $\kappa(n)$ on the KC is with respect to all verifiers, and when we want it only for honest ones we get the classes $\mathsf{SKC}^{\mathsf{type}}_{\text{hv}}[\kappa(n)]$ and $\mathsf{PKC}^{\mathsf{type}}_{\text{hv}}[\kappa(n)]$. Obviously $\mathsf{SKC}^{\mathsf{type}}[\kappa(n)] \subseteq \mathsf{SKC}^{\mathsf{type}}_{\text{hv}}[\kappa(n)]$ and $\mathsf{PKC}^{\mathsf{type}}[\kappa(n)] \subseteq \mathsf{PKC}^{\mathsf{type}}_{\text{hv}}[\kappa(n)]$. An obvious fact, but one to remember, is that the worst case KC is at most the average KC— $\mathsf{SKC}^{\text{wc}}[\,\kappa(n)\,] \subseteq \mathsf{SKC}^{\text{av}}[\kappa(n)]$ and $\mathsf{SKC}^{\text{wc}}_{\text{hv}}[\,\kappa(n)\,] \subseteq \mathsf{SKC}^{\text{av}}_{\text{hv}}[\kappa(n)]$.

DEFINITIONS. See Section 2 for formal definitions of existing classes, and Section 3 for our definition of oracle entropy.

DISCUSSION. As the above indicates, whether one considers knowledge which is zero or above, one has two kinds: statistical and perfect. The key ingredient in defining these is the simulator. In the zero-knowledge case it is a probabilistic, polynomial time machine [GMR]; to measure knowledge complexity, [GP] equip it with an oracle and then measure how much information the simulator gets from the oracle. When the output distribution of the simulator is exactly the view of a verifier the knowledge is perfect; when the two distributions have negligible statistical difference the knowledge is statistical. Different ways of measuring how much information the simulator is getting from its oracle result in the different types of KC mentioned above.

1.3 Statements of main results

THE PERFECT KC OF STATISTICAL ZK. We provide a novel characterization of statistical zero-knowledge as being exactly those languages leaking a negligible amount of perfect oracle entropy.

Theorem 1.1 $\mathsf{SZK} = \mathsf{PKC}^{\text{en}}[\,n^{-\omega(1)}\,]$.

We stress that this theorem holds for the full definitions where one considers cheating verifiers. But it also holds if one considers only honest verifiers; namely, $\mathsf{SZK}_{\text{hv}} = \mathsf{PKC}^{\text{en}}_{\text{hv}}[\,n^{-\omega(1)}\,]$. Intuitively, the theorem says that it is possible to produce a perfect simulation at the cost of a negligible amount of knowledge.

We also show that the SZK proofs leak only about one bit of perfect KC on the average.

Theorem 1.2 $\mathsf{SZK} \subseteq \mathsf{PKC}^{\text{av}}[\,1 + n^{-\omega(1)}\,]$.

Again, the same is true if one considers only honest verifiers; namely, $\mathsf{SZK}_{\text{hv}} \subseteq \mathsf{PKC}^{\text{av}}_{\text{hv}}[\,1 + n^{-\omega(1)}\,]$. Theorem 4.5 considers similar questions for knowledge complexity greater than zero. The proofs of Theorems 1.1 and 1.2 are in Section 4.

ORACLE ENTROPY. Recall that the intuition for measuring knowledge in the oracle framework of [GP] is that the knowledge leaked in a proof is measured by the information that a simulator needs from an oracle to provide a simulation. The strict and average KC measures of [GP] are obtained by measuring the oracle information by the number, and average number, of bits communicated to the simulator, respectively. We use instead an approach akin to Shannon's for defining the entropy of a source. Taking into account that the setting is computational (as reflected in the polynomiality of the simulator) we ask, akin to zero-knowledge, for an "oracle simulator," and measure its "success" in re-producing the simulator's output via entropy based measures. The definitions are in Section 3. The following theorem indicates that the difference between oracle entropy and the average measure

is at most an additive constant. Namely the oracle entropy of a language is never more than its average KC, and, on the other hand, can be less only by four.

Theorem 1.3 For any $\kappa(n) \leq \text{poly}(n)$–

- $\mathsf{PKC}^{\text{av}}[\kappa(n)] \subseteq \mathsf{PKC}^{\text{en}}[\kappa(n)] \subseteq \mathsf{PKC}^{\text{av}}[4+\kappa(n)]$
- $\mathsf{SKC}^{\text{av}}[\kappa(n)] \subseteq \mathsf{SKC}^{\text{en}}[\kappa(n)] \subseteq \mathsf{SKC}^{\text{av}}[4+\kappa(n)]$.

The proof is in Section 5. The same is true if one considers only honest verifiers, namely:

$$\mathsf{PKC}^{\text{av}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{PKC}^{\text{en}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{PKC}^{\text{av}}_{\text{hv}}[4+\kappa(n)]$$
$$\mathsf{SKC}^{\text{av}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{SKC}^{\text{en}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{SKC}^{\text{av}}_{\text{hv}}[4+\kappa(n)] \ .$$

Such theorems are important to indicate that we have not strayed too far from older measures.

Proofs of logarithmic KC. We show that under both measures languages of logarithmic KC are relatively simple.

Theorem 1.4 For any $\kappa(n) \leq O(\log n)$–

- $\mathsf{SKC}^{\text{av}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{BPP}^{\mathsf{NP}}$
- $\mathsf{SKC}^{\text{en}}_{\text{hv}}[\kappa(n)] \subseteq \mathsf{BPP}^{\mathsf{NP}}$.

Note that here considering the honest verifier classes strengthens the result since these classes are larger. Of course the containment also holds for the full case. The proof is in Section 6.

Expected versus strict poly-time simulation. Existing constructions of perfect ZK proofs attain the perfect ZK by allowing the simulator to run for expected, rather than strict, polynomial time, even though the verifier runs in strict polynomial time. Many researchers have noted that it seems intuitively more desirable for the simulator to run in strict polynomial time. To how much can we limit the knowledge lost?

Using techniques similar to those underlying the proof of Theorem 1.1 we can show that any perfect ZK proof with expected poly-time simulator has knowledge complexity in the oracle entropy sense at most $1/p(n)$, for any polynomial $p(n)$. (The simulator in the latter runs in strict poly-time, as for all our definitions.) An interesting open problem is whether this upper bound can be improved.

Protocols for set sizes. We provide constant round protocols for proving bounds on the size of sets to within much higher precision than that achieved by previous protocols. Our protocol uses higher independence. The analysis relies on a theorem of Maier about the distribution of primes in small intervals [M] and on the Bonferroni inequalities. See Section 8 for discussion of the motivations and the protocols.

1.4 Previous work

Our consideration of the perfect KC of statistical ZK was motivated by a result of [GOP] which implies that $\mathsf{SZK}_{\text{hv}} \subseteq \mathsf{PKC}^{\text{wc}}_{\text{hv}}[O(\log n)]$. As the notation indicates, this result is about the worst case measure but only for the honest verifier. Moving to oracle entropy and the average measure have provided two gains. The first, obviously, is the sizeable reduction in the perfect KC, from logarithmic to negligible (for oracle entropy) or about one bit (on the average). The second gain is that our results hold not only w.r.t. the honest verifier, but also for the full definition where cheating verifiers must be considered. Here nothing was known prior to our work— it was conceivable that a cheating verifier could extract a polynomial amount of perfect knowledge from a statistical ZK proof, under any measure.

As indicated above, Theorem 1.4 extends the result $\mathsf{SKC}^{\mathrm{wc}}_{\mathrm{hv}}[O(\log n)] \subseteq \mathsf{BPP}^{\mathsf{NP}}$ of [GOP], which in turn built on [BP]. Our proof will exploit several techniques underlying the previous works. In particular, the $\mathsf{BPP}^{\mathsf{NP}}$ complexity comes again from the use of the uniform generation procedure of [BP], and part of our analysis will extend an elegant argument of [GOP].

1.5 Discussion

Researchers have always considered perfect and statistical knowledge complexity as inherently different, reflected in the the two hierarchies corresponding to these notions. This dichotomy which treats knowledge lost and the imperfectness of the simulation as inherently different is in some ways non-intuitive. Theorems 1.1 and 4.5 provide a more unified view. They imply we need consider only one kind of KC, namely perfect; any deviation from perfection is measured *solely* in terms of knowledge complexity.

Statistical ZK is itself in some ways an "average" measure— the verifier does not learn nothing, but rather learns something only with negligible probability. Average KC is then perhaps a natural choice as the measure under which to investigate SZK.

Certainly some of the above theorems exemplify the ability to prove stronger results under the average measure than currently known under the worst case one. Meanwhile Theorem 1.4 indicates that central theorems about the worst case measure can extend to the average case; perhaps surprisingly, since, more often than not, results about worst case measures don't seem to extend to the average case.

2 Preliminaries

The statistical difference between random variables X, Y is

$$\mathrm{Dist}(X,Y) \;=\; \textstyle\sum_{\Lambda\in\{0,1\}^*} \left|\Pr[\,X=\Lambda\,]-\Pr[\,Y=\Lambda\,]\right| \;.$$

PPT stands for probabilistic, polynomial time. If PPT machine $A(\cdot,\cdots)$ takes inputs $x_1,\ldots$ then $A(x_1,\ldots)$ denotes the random variable attaining value v with the probability that A, on inputs $x_1,\ldots$, outputs v. Often we need to make the coin tosses of A explicit; we write $A(x_1,\ldots;R)$ for the output of A on inputs $x_1,\ldots$, and coins R. Note by definition

$$\Pr[\,A(x_1,\ldots)=v\,] \;=\; \Pr_R[\,A(x_1,\ldots;R)=v\,] \;.$$

Interactive Proofs. A verifier is a PPT machine, and a prover is a probabilistic function. They interact on a common input usually denoted x, with n denoting its length. If P is a prover and V is a verifier, we say that (P,V) is an interactive proof for L with error $\epsilon(\cdot)$ if the following are true:

(1) *Completeness.* If $x \in L$ then P can convince V to accept with probability at least $1-\epsilon(n)$.

(2) *Soundness.* If $x \notin L$ then for all provers P', the probability that P' can convince V to accept is at most $\epsilon(n)$.

Typically in a knowledge complexity setting one says that (P,V) is a proof system if it has negligible error. This is important for Theorem 1.4. But Theorems 1.1 and 1.2 don't need this restriction: the transformations underlying these theorems change the error by at most a negligible additive term, so any reasonable convention on what to call a proof may be used.

Views and ZK. The view of a (possibly cheating) verifier V' in its interaction with prover P on input x, denoted $\mathrm{View}^P_{V'}(x)$, is a random variable taking values which are pairs $(R, \alpha_1\beta_1\ldots\alpha_g\beta_g)$, the first coordinate representing the coins of V' and the second the conversation between V' and

P when the former's coins are set to R. A simulator is a PPT machine S, which, following [GP], is required to produce an output with probability at least $1/2$, and otherwise indicate failure by outputting a special symbol fail. For simulators we adopt the special convention that $S(x)$ denotes the random variable defining the output conditioned on the good event that an output is produced; in other words, if G is the subset of the simulators coins on which simulator produces an output, then $\Pr[\, S(x) = \Lambda \,]$ denotes the probability that the simulator produces output Λ when its coins R are chosen randomly from G. We say that S is a statistical (resp. perfect) ZK (P, V') simulator if $\mathrm{Dist}(\mathrm{View}^P_{V'}(x), S(x))$ is at most $n^{-\omega(1)}$ (resp. is zero). An interactive proof (P, V) for L is SZK (resp. PZK) if for every verifier V' there is a statistical (resp. perfect) ZK (P, V') simulator.

The oracle framework of KC. In the knowledge complexity setting, the simulator is allowed access to an oracle [GP]. It is instructive to make the discussion of oracle interaction independent of the setting; in particular this will help when we discuss oracle entropy. So consider a PPT machine $A(\cdot)$ which takes an input x and has access to an oracle γ. The latter takes a string of any length. We adopt the convention that it may return any *non-empty* string. On coins R and input x, machine A will perform some computation, in the course of which it will query the oracle γ, perhaps adaptively and several times, to eventually produce an output $A^\gamma(x; R)$. Let $C_x(R)$ be the number of bits communicated from γ to A in the execution $A^\gamma(x; R)$. (Ie. the sum, over all oracle queries, of the number of bits returned in answer to that query.) Then we write $\mathbf{Com}_L^{\gamma \to A}(n) \leq \kappa(n)$ if for all $x \in L$ and all R it is the case that $C_x(R) \leq \kappa(n)$. We write $\mathbf{AvCom}_L^{\gamma \to A}(n) \leq \kappa(n)$ if for all $x \in L$ it is the case that $\mathbf{E}_R\,[\, C_x(R) \,] \leq \kappa(n)$.

Sometimes the oracle is allowed to be probabilistic: this means $\gamma(z)$ is a distribution on the set of non-empty strings, and a random element of this distribution is returned on query z.

Knowledge complexity. Now S^γ is a statistical (resp. perfect) (P, V') simulator if the statistical difference between $\mathrm{View}^P_{V'}(x)$ and $S^\gamma(x)$ is at most $n^{-\omega(1)}$ (resp. is zero). It has knowledge complexity $\kappa(n)$ in the strict (resp. average) oracle sense if $\mathbf{Com}_L^{\gamma \to S}(n) \leq \kappa(n)$ (resp. $\mathbf{AvCom}_L^{\gamma \to S}(n) \leq \kappa(n)$). As usual an interactive proof (P, V) for L is said to have SKC (resp. PKC) in a particular sense if each verifier V' possesses a SKC (resp. PKC) $\kappa(n)$ simulator in this sense. The notation for the complexity classes corresponding to all these definitions was provided in Section 1.2.

On conventions about oracles. The formal definition in [GP] asks that the oracle answers be bits. Here we allow any non-empty string. Allowing only bits seems restrictive. For example, suppose the oracle is returning an integer i whose length we don't know à priori. Under our convention, the number of bits returned is the length of i, which is seems intuitively what one would want. But if i must be returned a bit at a time then it must be properly encoded, for example by having the oracle also specify its length, and this adds $\log(|i|)$ bits to the communication.

One might think that if answers of arbitrary length are allowed then one needs only a single oracle query consisting of x, R, in answer to which all necessary bits are returned. But this is not a good convention. For example it would then become unclear how to prove something that seems intuitve: that the KC (say in the average sense) should be additive under sequential composition. Thus it is still important to allow many queries.

Finally, we note that these problems with encoding seem very model dependent, and illustrate another advantge of oracle entropy. In the latter approach, one measures only the information content of the oracle's reply, so that the manner in which the communication is encoded does not affect the knowledge complexity.

3 Oracle entropy

As previous discussions have indicated, the motivation that [GP] provided for the oracle framework is that it provides a "reduction:" the task of measuring how much "knowledge" P gave V' is reduced to the task of measuring how much "information" S needed from γ in order to reproduce the view of V'. Hopefully, the latter measuring task is simpler than the former, due to the simpler setting. In this light, [GP] have chosen the simplest measures of the amount of information provided by the oracle, namely the number of bits communicated, or the average of this, as defined above. But this can be an unnecessary over-count. (For example, under this measure, whether a simulator got $\kappa(n)$ zeros or $\kappa(n)$ bits of output of a hard function makes no difference; the number of bits of information is $\kappa(n)$). In particular, this measure is not adequate to prove theorems like Theorem 1.1. Thus we suggest a new definition of the amount of information that an oracle provides to a machine using this oracle. Since it is applicable in other contexts such as PCP, the discussion is in the general setting of a machine talking to an oracle discussed above.

Entropy based measure of information provided by an oracle. We consider a PPT machine $A(\cdot)$ which tosses $r(\cdot)$ coins. Its input x has length n and its coin tosses are denoted by R. It has access to an oracle γ. (For simplicity we restrict ourselves in the following to deterministic oracles. Probabilistic ones can be handled similarly. See Appendix A). Its output is $A^{\gamma}(x;R)$. We wish to measure how many bits of information A got from γ. An upper bound on the number of bits of information that A got is certainly the total number of bits returned by the oracle over the course of the execution. However, as already discussed, this may be not be a good bound, and in particular does not capture information in the sense of its "use" to the polynomially bounded A. We suggest a more information theoretic definition, as follows. The goal of A is to produce its output value $v = A^{\gamma}(x;R)$. We measure the amount of information A got from γ, in its run on x, R, in terms of the probability that, on these inputs, one could efficiently produce v without access to γ.

In order to do this we consider a probabilistic machine $A'(\cdot,\cdot)$ which we call an *oracle simulator*. It tosses some $r'(\cdot)$ coins. It is given x, R but not γ, and tries to produce v. For $R \in \{0,1\}^{r(n)}$ define

$$P_x(R) \;=\; \Pr_{\rho \leftarrow \{0,1\}^{r'(n)}}[\, A'(x,R;\rho) = A^{\gamma}(x;R)\,] \;.$$

We stress that the probability in the definition of $P_x(R)$ is over the coin tosses of A', for each fixed x, R. Now define the "information provided to A by γ on input x, as measured by A'," to be:

$$\mathbf{Info}_{A'}^{\gamma \to A}(x) \;=\; \mathbf{E}_{R \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{r(n)}} \left[\lg \frac{1}{P_x(R)} \right] .$$

Definition 3.1 We write $\mathbf{Info}_{L}^{\gamma \to A}(n) \leq \kappa(n)$ if there is an oracle simulator A' such that $\mathbf{Info}_{A'}^{\gamma \to A}(x) \leq \kappa(|x|)$ for all $x \in L$. We say that A gets at most $\kappa(n)$ bits of information from γ, in the entropy sense, if $\mathbf{Info}_{L}^{\gamma \to A}(n) \leq \kappa(n)$.

We stress that showing $\mathbf{Info}_{L}^{\gamma \to A}(n) \leq \kappa(n)$ involves exhibiting a new machine, namely the oracle simulator A', just as showing a protocol is ZK involves exhibiting a new machine, namely the simulator.

The Oracle entropy measure of KC. The above is easily translated into the setting in which we are interested, by setting A^{γ} to be the simulator S^{γ}. Thus a simulator S^{γ} for (P, V') is said to have KC $\kappa(n)$ in the oracle entropy sense if $\mathbf{Info}_{L}^{\gamma \to S}(n) \leq \kappa(n)$. An interactive proof (P, V) for L is said to have statistical (resp. perfect) oracle entropy $\kappa(n)$ if each verifier V' possesses a statistical

(resp. perfect) simulator of KC $\kappa(n)$ in the oracle entropy sense. The notation for the complexity classes corresponding to these definitions was provided in Section 1.2.

We note that showing (P, V) has oracle entropy $\kappa(n)$ involves exhibiting for each verifier V' three things: a PPT machine S; an oracle γ; and an oracle simulator.

4 Perfect versus statistical

We prove Theorems 1.1 and 1.2, and conclude by stating some extensions to KC greater than zero.

We are given a SZK interactive proof (P, V) for L. Recall that P is a probabilistic function. It takes as input x and a partial conversation $\alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}\alpha_t$, and produces the next message $\beta = \beta_t$ which is sent to the verifier. Arbitrary provers, such as the interactive proof setting considers, can produce messages under arbitrary distributions. (For example, there exist statistical ZK proofs in which P sends a particular string with probability exactly $1/\pi$). Recall that our goal is to achieve a *perfect* simulation of an interaction. But trying to reach such strange distributions causes difficulties. Thus our first step is to modify the prover to be what we call *samplable*. We assume that all messages are of length $l(n)$ for some polynomial $l(n)$.

Definition 4.1 Let P be a prover with message length $l(n)$. We say that P is samplable if there is a polynomial $p(n)$ such that for every x, every conversation prefix $\alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}\alpha_t$, and every $\beta \in \{0,1\}^{l(n)}$ there is an integer $w \in \mathsf{N}$ such that $\Pr[\, P(x, \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}\alpha_t) = \beta \,] = w \cdot 2^{-p(n)}$.

The following is a simple fact about probability distributions, noted also in [GOP].

Lemma 4.2 Suppose (P_1, V) is a SZK proof for L with error $\epsilon(n)$. Then there exists a samplable prover P_2 such that (P_2, V) is a SZK proof for L with error $\epsilon(n) + n^{-\omega(1)}$.

The main lemma is the one that follows. It considers a samplable prover P. If (P, V) is statistical zero-knowledge, then, for any verifier V', there exists a statistical ZK (P, V')-simulator. Focusing on any such verifier, we show that there exists another simulator who can produce a *perfect* simulation of the real interaction, with appropriately low KC in the oracle entropy and average senses. The proof is in Section 4.1 below.

Lemma 4.3 (Probability Shifting Lemma) Let P be a samplable prover and V' a verifier. Suppose there exists a statistical ZK (P, V') simulator. Then there exists a perfect (P, V') simulator with KC $n^{-\omega(1)}$ in the oracle entropy sense and $1 + n^{-\omega(1)}$ in the average sense.

Theorem 1.2 follows from Lemmas 4.3 and 4.2, as does one half of Theorem 1.1, namely $\mathsf{SZK} \subseteq \mathsf{PKC}^{\mathrm{en}}[n^{-\omega(1)}]$. The opposite containment is implied by the following lemma, which completes the proof of Theorem 1.1. Its proof is in Section 4.2 below.

Lemma 4.4 Let (P, V) be a $\mathsf{PKC}^{\mathrm{en}}[n^{-\omega(1)}]$ proof for L. Then (P, V) is a SZK proof for L.

Extensions to higher KC. The following extend Theorems 1.1 and 1.2 by showing the cost of making statistical KC into perfect when the former is non-zero.

Theorem 4.5 For any $\kappa(n) \leq \mathrm{poly}(n)$–

$$\mathsf{SKC}^{\mathrm{en}}\,[\,\kappa(n)\,] \subseteq \mathsf{PKC}^{\mathrm{en}}\,[\,\kappa(n) + n^{-\omega(1)}\,]$$
$$\mathsf{SKC}^{\mathrm{av}}\,[\,\kappa(n)\,] \subseteq \mathsf{PKC}^{\mathrm{av}}\,[\,\kappa(n) + 1 + n^{-\omega(1)}\,]\,.$$

The following feature of the transformations here will be useful later:

Remark 4.6 The transformations underlying Theorem 4.5 preserve the probability with which the simulator produces an output. Also if the oracle for the given simulator is deterministic then so is the oracle for the final one.

As usual the theorem is true also for the honest verifier case. In fact, the honest verifier case of the second statement in the theorem, and the remark, will be used in our proof of Theorem 1.4.

We note that unlike Theorem 1.1 the first statement in Theorem 4.5 is a containment, not a characterization. We have not looked closely at the problem of whether it can be made an equality, and this remains an open question. The proofs of the above are similar to those of Theorems 1.1 and 1.2 and are omitted from this abstract.

Combining the above theorem and Theorem 1.3 will yield a variety of containments relating various kinds of statistical and perfect KC. These statements are omitted here.

4.1 Proof of Lemma 4.3

We will define a perfect (P, V') simulator S^{γ} having the claimed KC. Let S_1 be a statistical ZK (P, V') simulator and let $r(n)$ be the number of coins it flips. Let $x \in L$. Let $D = [\mathrm{View}^P_{V'}(x)] \cup [S(x)]$ be the views that arise in either the real interaction or the simulation. Fix a polynomial $\nu(n)$ such that each $\Lambda \in D$ has length exactly $\nu(n)$. Also order the elements of D in some canonical view, writing them as $\Lambda_1, \ldots, \Lambda_N$.

We know that P is samplable, V' is poly(n) time, and S_1 is poly(n) time; using this, we can show that there is a polynomial $p(n) \geq r(n)$ such for any $i \in [N]$ there exists a pair of integers $w(i), w_1(i) \in \mathsf{N}$ such that:

$$\begin{aligned} \Pr[\,\mathrm{View}^P_{V'}(x) = \Lambda_i\,] &= w(i) \cdot 2^{-p(n)} \\ \Pr[\,S_1(x) = \Lambda_i\,] &= w_1(i) \cdot 2^{-p(n)}\ . \end{aligned}$$

Appending extra coins to the random tape of S_1 won't change anything, so we may assume wlog that $r(n) = p(n)$. Let

$$\begin{aligned} A_1 &= \{\,(i,j) \,:\, i \in [N] \text{ and } w(i) < j \leq w_1(i)\,\} \\ A &= \{\,(i,j) \,:\, i \in [N] \text{ and } w_1(i) < j \leq w(i)\,\}\ . \end{aligned}$$

Let $I^+ = \{\, i \in [N] \,:\, w_1(i) > w(i)\,\}$ and $I^- = \{\, i \in [N] \,:\, w_1(i) < w(i)\,\}$. The following equalities, which say that the positive and negative statistical differences between $\mathrm{View}^P_{V'}(x)$ and $S_1(x)$ are the same, show that A_1 and A have the same size.

$$\begin{aligned} |A_1| &= \textstyle\sum_{i \in I^+} (\, w_1(i) - w(i)\,) \\ &= \textstyle\sum_{i \in I^-} (\, w(i) - w_1(i)\,) \\ &= |A|\ . \end{aligned} \tag{1}$$

Thus there exists a bijection $f\colon\ A_1 \to A$. Fix such a bijection, and let $f_1\colon\ A_1 \to [N]$ be its first coordinate. Let $\mathcal{R}$ be the set of good random taps of S_1 (namely those on which S_1 does not output fail).

Specification of the oracle. The oracle γ takes as input x and a random tape $R \in \mathcal{R}$ of S_1. It computes $(i,j) = \mathrm{Ind}(R)$. If $j \leq w(i)$ then it returns the bit 1. Else, it returns the bit 0, followed by the $\nu(n)$-bit string $\Lambda_{f_1(i,j)}$.

Specification of S. The machine S has inputs x and oracle access to the oracle γ defined above. It picks at random a random tape $R \in \{0,1\}^{p(n)}$ for S_1. It first runs $S_1(x; R)$ to see if it fails; if

so, so does S, and this happens with probability at most one-half. If there is an output Λ, then S poses the query x, R of its oracle γ. If the response is the bit 1 then S outputs Λ. Else the response from the oracle was (the bit 0 followed by) some view Λ', and S outputs Λ'.

We must show that this is a *perfect* simulation.

Lemma 4.7 Let $x \in L$. Then $S^\gamma(x) = \mathrm{View}^P_{V'}(x)$.

Proof: Fix $k \in [N]$. We claim that

$$\Pr_R[\, S^\gamma(x;R) = \Lambda_k \,] \;=\; w(k) \cdot 2^{-p(n)} \;. \tag{2}$$

The left side is $\Pr[\, S^\gamma(x) = \Lambda_k \,]$ and the right side is $\Pr[\, \mathrm{View}^P_{V'}(x) = \Lambda_k \,]$ so this will conclude the proof. We now establish (2).

If $w(k) \le w_1(k)$ then (2) is clear from the definitions of S and γ. So suppose $w_1(k) < w(k)$. Then

$$\Pr_R[\, S^\gamma(x;R) = \Lambda_k \,] \;=\; \frac{w_1(k) + |\{\, (i,j) \in A_1 \,:\, f_1(i,j) = k \,\}|}{2^{p(n)}} \;.$$

But f is a bijection. So

$$\begin{aligned} |\{\, (i,j) \in A_1 \,:\, f_1(i,j) = k \,\}| &= |\{\, (i,j) \in A \,:\, i = k \,\}| \\ &= w(i) - w_1(i) \;. \end{aligned}$$

Putting these equations together concludes the proof. ∎

Note the strings sent by the oracle are sometimes of polynomial length $\nu(n)$. However it is still easy to see that on the average, S gets only $1 + n^{-\omega(1)}$ bits from its oracle. The proof that S gets only a negligible amount KC in the oracle entropy sense is more interesting.

Lemma 4.8 $\mathbf{Info}^{\gamma \to S}_L(n) = n^{-\omega(1)}$.

Proof: We exhibit an oracle simulator S'. Its inputs are x and a random tape $R \in \{0,1\}^{p(n)}$ of S. It first runs the *original* simulator S_1 on input x and coins R. If the output is fail then S' outputs fail. Else it denotes the output by Λ. Now it flips a coin which is 1 with probability $q = 2^{-n}$ and 0 otherwise. If the coin is 1 then it outputs a random string of length $\nu(n)$. Else it outputs Λ.

Let $P_x(R) = \Pr[\, S'(x,R) = S^\gamma(x;R) \,]$. (By definition the probability is over the coin tosses of S'). Note it is 1 when $R \notin \mathcal{R}$, and thus the contribution to $\mathbf{Info}^{\gamma \to S}_{S'}(x)$ is 0 in this case. So we concentrate on $R \in \mathcal{R}$. In the following, probabilities and expectations are over R chosen at random from $\mathcal{R}$. Now let $X_1 = \{\, (i,j) \,:\, i \in [N] \text{ and } 1 \le j \le w_1(i) \,\}$. We may fix a bijection $\mathrm{Ind}\colon \mathcal{R} \to X_1$. Let $A'_1 = \{\, (i,j) \,:\, i \in [N] \text{ and } 1 \le j \le \min(w(i), w_1(i)) \,\}$. Note $X_1 = A_1 \cup A'_1$ is a partition of X_1. Also, by Equation (1) we know that $\Pr_R[\, \mathrm{Ind}(R) \in A_1 \,]$ is negligible.

The analysis for $R \in \mathcal{R}$ breaks into two cases, according to $(i,j) = \mathrm{Ind}(R)$. The first case is that $w(i) < j$, where the output of S' is correct with probability at least $1 - q$. The second case is if $w(i) \ge j$, where the probability that the output of S' is correct is at least $q \cdot 2^{-\nu(n)}$. Thus

$$\begin{aligned} \mathbf{Info}^{\gamma \to S}_{S'}(x) \;\le\; \mathbf{E}_R\left[\lg \frac{1}{P_x(R)} \right] &\le \Pr_R[\, \mathrm{Ind}(R) \in A'_1 \,] \cdot \lg \frac{1}{1-q} \;+\; \Pr_R[\, \mathrm{Ind}(R) \notin A'_1 \,] \cdot \lg \frac{1}{q \cdot 2^{-\nu(n)}} \\ &\le \lg \frac{1}{1 - 2^{-n}} \;+\; \Pr_R[\, \mathrm{Ind}(R) \in A_1 \,] \cdot \lg 2^{n+\nu(n)} \;. \end{aligned}$$

But this is at most $2^{-n+1} + n^{-\omega(1)} \cdot \mathrm{poly}(n) \le n^{-\omega(1)}$ ∎

4.2 Proof of Lemma 4.4

Let S be a perfect (P, V') simulator such that $\mathbf{Info}_L^{\gamma \to S}(n) \leq n^{-\omega(1)}$. We construct a SZK (P, V') simulator S_1.

We will assume γ is deterministic and use the definitions of Section 2— The case of probabilistic γ is left to the final paper. For simplicity also disregard the technicality that S produces an output only with probability $1/2$, and assume it always produces an output— the extension is easily dealt with.

By assumption there is an oracle simulator S' such that $\mathbf{Info}_{S'}^{\gamma \to S}(x) \leq n^{-\omega(1)}$ for all $x \in L$. Let $r(n)$ be the number of coins tossed by S and $r'(n)$ the number of coins tossed by S'. We define S_1 as follows. On inputs x, it picks $R \in \{0,1\}^{r(n)}$ at random and $\rho \in \{0,1\}^{r'(n)}$ at random. It runs $S'(x, R; \rho)$ to get an output Λ, and it outputs Λ.

Let $P_x(R) = \Pr_\rho[\, S'(x, R; \rho) = S^\gamma(x; R)\,]$. By assumption there is a function $\mu(n) = n^{-\omega(1)}$ such that $\mathbf{Info}_L^{\gamma \to S}(n) \leq \mu(n)$.

Claim 4.9 Let $x \in L$. Then

$$\mathbf{E}_{R \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{r(n)}} [\, 1 - P_x(R)\,] \;\leq\; \mu(n)\,.$$

Proof: The inequality $y \leq -\lg(1-y)$ is valid for all $y \in [0,1]$. So

$$\mathbf{E}_R [\, 1 - P_x(R)\,] \;\leq\; \mathbf{E}_R [\, -\lg P_x(R)\,]\,.$$

But the latter is by assumption at most $\mu(n)$. ∎

For Λ a view in the range of $\mathrm{View}_{V'}^P(x)$ let $T(\Lambda) = \{\, R \,:\, S^\gamma(x; R) = \Lambda\,\}$. Also let $P(R, \Lambda) = \Pr_\rho[\, S'(x, R; \rho) = \Lambda\,]$. Use Claim 4.9 to note the following:

$$\begin{aligned} \textstyle\sum_\Lambda \sum_{R \notin T(\Lambda)} 2^{-r} \cdot P(R, \Lambda) &= \textstyle\sum_R \sum_{\Lambda \neq S^\gamma(x;R)} 2^{-r} \cdot P(R, \Lambda) \\ &= \textstyle\sum_R 2^{-r} \cdot (1 - P_x(R)) \\ &\leq \mu(n)\,. \end{aligned} \tag{3}$$

We now bound the statistical difference between $\mathrm{View}_{V'}^P(x)$ and $S_1(x)$. We have

$$\begin{aligned} &\textstyle\sum_\Lambda \left| \Pr_R[\, \mathrm{View}_{V'}^P(x) = \Lambda\,] - \Pr_{R,\rho}[\, S_1(x; R, \rho) = \Lambda\,] \right| \\ &\quad = \textstyle\sum_\Lambda \left| \Pr_R[\, S^\gamma(x; R) = \Lambda\,] - \Pr_{R,\rho}[\, S'(x, R; \rho) = \Lambda\,] \right| \\ &\quad = \textstyle\sum_\Lambda 2^{-r} \cdot \left| |T(\Lambda)| - \sum_{R \in T(\Lambda)} P(R, \Lambda) - \sum_{R \notin T(\Lambda)} P(R, \Lambda) \right|\,. \end{aligned}$$

Note $P(R, \Lambda) = P_x(R)$ when $R \in T(\Lambda)$. Then apply the triangle inequality to bound the above by:

$$\textstyle\sum_\Lambda 2^{-r} \cdot \left| |T(\Lambda)| - \sum_{R \in T(\Lambda)} P_x(R) \right| \;+\; \sum_\Lambda \sum_{R \notin T(\Lambda)} 2^{-r} \cdot P(R, \Lambda)\,.$$

By Equation (3), the second sum is at most $\mu(n)$. On the other hand we can bound the first sum by

$$\textstyle\sum_\Lambda 2^{-r} \cdot \sum_{R \in T(\Lambda)} (1 - P_x(R)) \;=\; 2^{-r} \cdot \sum_R (1 - P_x(R))\,.$$

This last quantity is at most $\mu(n)$ by Claim 4.9. So the statistical difference in question is at most $2\mu(n) = n^{-\omega(1)}$, which completes the proof.

5 Proof of Theorem 1.3

We first dispense with the simple direction and then move on to the more interesting one.

5.1 $\mathsf{PKC}^{\mathrm{av}}[\kappa(n)] \subseteq \mathsf{PKC}^{\mathrm{en}}[\kappa(n)]$

Let (P,V) be a $\mathsf{PKC}^{\mathrm{av}}[\kappa(n)]$ proof for L. We claim that (P,V) is a $\mathsf{PKC}^{\mathrm{en}}[\kappa(n)]$ proof for L. To show this, let V' be any verifier, and let S^{γ} be a (P,V')-simulator such that $\mathbf{AvCom}_L^{\gamma\to S}(n) \leq \kappa(n)$. It suffices to show that $\mathbf{Info}_L^{\gamma\to S}(n) \leq \kappa(n)$. We present an appropriate oracle simulator S'. On inputs x, R, machine S' runs S on inputs x and with coins R, and provides random responses to oracle queries of S. By definition, S on inputs x and coins R will receive $C_x(R)$ bits from γ, which means that the answers provided by S' are correct with probability at least $2^{-C_x(R)}$. Thus $P_x(R)$ is at least $2^{-C_x(R)}$. Now we can see that

$$\mathbf{Info}_{S'}^{\gamma\to S}(x) \;=\; \mathbf{E}_R\left[\lg\frac{1}{P_x(R)}\right] \;\leq\; \mathbf{E}_R\left[\lg\frac{1}{2^{-C_x(R)}}\right] \;=\; \mathbf{E}_R\left[\,C_x(R)\,\right].$$

However the last quantity is by definition $\mathbf{AvCom}_L^{\gamma\to S}(n)$, and by assumption this is at most $\kappa(n)$.

5.2 $\mathsf{PKC}^{\mathrm{en}}[\kappa(n)] \subseteq \mathsf{PKC}^{\mathrm{av}}[4+\kappa(n)]$

As in Section 4.2 disregard, for simplicity, the technicality of simulators only producing output with probability 1/2, and assume they always produce an output— the extension is easily dealt with.

Let (P,V) be a $\mathsf{PKC}^{\mathrm{en}}[\kappa(n)]$ proof for L. We claim that (P,V) is a $\mathsf{PKC}^{\mathrm{av}}[4+\kappa(n)]$ proof for L. To show this, let V' be any verifier, and let S^{γ} be a perfect (P,V')-simulator such that $\mathbf{Info}_L^{\gamma\to S}(n) \leq \kappa(n)$. We present a perfect (P,V') simulator $S_1^{\gamma_1}$ such that $\mathbf{AvCom}_L^{\gamma_1\to S_1}(n) \leq 4+\kappa(n)$. Our construction uses universal hash functions and is influenced by the proof of [GP, Proposition 3.2].

By assumption there is an oracle simulator S' such that $\mathbf{Info}_{S'}^{\gamma\to S}(x) \leq \kappa(n)$ for all $x \in L$. Let $r'(n)$ be the number of coins it tosses, and let $r(n)$ be the number of coins tossed by S. For each $x \in L$ and each $R \in \{0,1\}^{r(n)}$ let

$$\begin{aligned} G_x(R) &= \{\,\rho \in \{0,1\}^{r'(n)} \,:\, S'(x,R;\rho) = S^{\gamma}(x;R)\,\} \\ P_x(R) &= |G_x(R)| \cdot 2^{-r'(n)}\,. \end{aligned}$$

Note that $P_x(R) > 0$, whence $G_x(R) \neq \emptyset$; this follows from the fact that $\mathbf{E}_R[\,\lg(1/P_x(R))\,] \leq \kappa(n) < \infty$.

We let $l(n) = n + r'(n)$ and $L = 2^l$. We fix a collection H_n of pairwise independent hash functions mapping $[L] \stackrel{\mathrm{def}}{=} \{0,\ldots,L-1\}$ to $\{0,1\}^{r'(n)}$. We now describe the new simulator $S_1^{\gamma_1}$.

The oracle γ_1. Oracle γ_1 will receive as input x, R and a hash function $h \in H_n$. It lets $S_h = \{\, i \in [L] \,:\, h(i) \in G_x(R)\,\}$ and responds as follows:

(1) If $S_h \neq \emptyset$ then γ_1 returns $0\,.\,i$ where i is the least element of S_h. (Here i is an integer, interpreted as a string by writing it in binary.)

(2) If $S_h = \emptyset$ then γ_1 returns $1\,.\,\rho$ where ρ is the least element of $G_x(R)$.

The machine S_1. On input x, the machine S_1 picks at random a random tape $R \in \{0,1\}^{r(n)}$ for S. It also picks at random a hash function h from H_n. We let $S_h = \{\, i \in [L] \,:\, h(i) \in G_x(R)\,\}$. S_1 will provide the oracle γ_1 with x, R, h, and depending on the response of the oracle, it behaves as follows:

(1) If the response of γ_1 has the form $0\,.\,i$ then it sets $\rho = h(i) \in \{0,1\}^{r'(n)}$, computes $\Lambda = S'(x, R; \rho)$, and outputs Λ.

(2) Else if the response of γ_1 has the form $1\,.\,\rho$ then it computes $\Lambda = S'(x, R; \rho)$ and outputs Λ.

We have to show two things: that the simulation of (S_1, γ_1) is correct, and that on the average, the number of bits that S_1 gets from γ_1 is at most $4 + \kappa(n)$.

Lemma 5.1 Let $x \in L$. Then $S_1^{\gamma_1}(x) = \mathrm{View}_{V'}^{P}(x)$.

Proof: The output of S_1 is $S'(x, R; \rho)$ for a string ρ which is always in the set $G_x(R)$. The latter condition guarantees that $S'(x, R; \rho) = S^{\gamma}(x; R)$. Moreover R is chosen at random, and we know that $S^{\gamma}(x) = \mathrm{View}_{V'}^{P}(x)$. ∎

The random choices of S_1 are R and h. The number of bits communicated from γ_1 to S_1 on inputs x and coins $R\,.\,h$ is denoted $C_x(R\,.\,h)$. We are interested in $\mathbf{E}_R\mathbf{E}_h\,[\,C_x(R\,.\,h)\,]$. Let us first fix R and analyze $\mathbf{E}_h\,[\,C_x(R\,.\,h)\,]$.

Lemma 5.2 Let $x \in L$ and $R \in \{0,1\}^{r(n)}$. Then

$$\mathbf{E}_h\,[\,C_x(R\,.\,h)\,] \;\le\; 3 + \left\lceil \lg \frac{1}{P_x(R)} \right\rceil \;.$$

Proof: For each $i \in [L]$ let the random variable X_i be 1 if $h(i) \in G_x(R)$ and 0 otherwise. For $m = 1, \ldots, L$ let $S_m = X_0 + \cdots + X_{m-1}$. The random variables $X_0, \ldots, X_{L-1}$ are pairwise independent, each with expectation $P_x(R)$. So $\mathbf{E}\,[S_m] = m \cdot P_x(R)$, and by Chebyshev:

$$\Pr[\,S_m = 0\,] \;\le\; \Pr[\,|S_m - \mathbf{E}\,[S_m]| \ge \mathbf{E}\,[S_m]\,] \;\le\; \frac{\mathbf{E}\,[S_m]}{\mathbf{E}\,[S_m]^2} \;=\; \frac{1}{m \cdot P_x(R)} \;. \tag{4}$$

Define the random variable M to be

$$\begin{cases} \min(i \in [L] \;:\; X_i = 1) & \text{if } S_L > 0 \\ L - 1 & \text{otherwise.} \end{cases}$$

If Case (1) in the definition of γ_1 occurs then $C_x(R\,.\,h)$ is $1 + \cdot\mathrm{bin}(M)$, where $\mathrm{bin}(\cdot)$ denotes the length of the binary representation of an integer. But our choice of l guarantees that $r'(n) \le l = \mathrm{bin}(L-1)$, so $1 + \cdot\mathrm{bin}(M)$ is also a bound in the second case. Thus $C_x(R) \le 1 + \cdot\mathrm{bin}(M)$. So we need to estimate $\mathbf{E}_h\,[\,\mathrm{bin}(M)\,]$. Now let $N = \lceil \lg(1/P_x(R)) \rceil$. Then

$$\begin{aligned}
\mathbf{E}_h\,[\,\mathrm{bin}(M)\,] &= \Pr[\,0 \le M < 2\,] \cdot 1 \;+ \textstyle\sum_{j=2}^{l} \Pr[\,2^{j-1} \le M < 2^j\,] \cdot j \\
&= 1 \;+ \textstyle\sum_{j=2}^{l} \Pr[\,2^{j-1} \le M < 2^j\,] \cdot (j-1) \\
&= 1 \;+ \textstyle\sum_{j=1}^{l-1} \Pr[\,2^{j} \le M < 2^l\,] \\
&\le 1 + N + \textstyle\sum_{j=N+1}^{l-1} \Pr[\,M \ge 2^j\,] \\
&= 1 + N + \textstyle\sum_{j=1}^{l-N-1} \Pr[\,M \ge 2^{j+N}\,] \;.
\end{aligned}$$

But $2^N \ge 1/P_x(R)$. Now using Equation (4) the above is at most

$$\begin{aligned}
1 + N + \textstyle\sum_{k=1}^{l-N-1} \Pr[\,M \ge 2^k/P_x(R)\,] &\le 1 + N + \textstyle\sum_{k=1}^{l-N-1} \Pr[\,S_{\lceil 2^k/P_x(R) \rceil} = 0\,] \\
&\le 1 + N + \textstyle\sum_{k=1}^{l-N-1} \dfrac{1}{(2^k/P_x(R)) \cdot P_x(R)} \\
&= 1 + N + \textstyle\sum_{k=1}^{l-N-1} 2^{-k} \\
&\le 2 + N \;.
\end{aligned}$$

This proves the lemma. ∎

Now we can easily complete the estimation of the expected communication:

$$\begin{aligned}
\mathbf{AvCom}_L^{\gamma_1 \to S_1}(n) &= \mathbf{E}_R \mathbf{E}_h \left[C_x(R \,.\, h) \right] \\
&\leq \mathbf{E}_R \left[3 + \left\lceil \lg \frac{1}{P_x(R)} \right\rceil \right] \\
&\leq 4 + \mathbf{E}_R \left[\lg \frac{1}{P_x(R)} \right] \\
&= 4 + \mathbf{Info}_{S'}^{\gamma \to S}(x) \\
&\leq 4 + \kappa(n) \,.
\end{aligned}$$

This completes the proof.

6 Proof of Theorem 1.4

We are given $L \in \mathsf{SKC}^{\mathrm{av}}_{\mathrm{hv}}[O(\log n)]$ and we want to show $L \in \mathsf{BPP}^{\mathsf{NP}}$.

Some notation and simplifying assumptions. We fix an interactive proof (P, V) for L with a negligible error probability denoted $\epsilon(n)$. We fix S^γ, a *perfect* simulator for (P, V) with KC $\kappa(n) = O(\log n)$ in the average sense. We assume that S^γ produces outputs with probability one rather than one-half, and also that γ is deterministic. Finally, we assume the oracle returns its answers a bit it a time; ie. the answer to any oracle query is a single bit, and S queries the oracle $\kappa(n)$ times on the average.

The assumptions are wlog. Let's see how to get into the above situation. We are given (P_1, V) an interactive proof for L with negligible error, and $S_1^{\gamma_1}$ a statistical simulator for (P_1, V) with KC $O(\log n)$ in the average sense. We transform the system so that the above is true. We first simplify the simulator in two ways. First, it can be made to produce an output with probability one, rather than one-half, by [GP, Proposition 3.8]. Second, the oracle can be made deterministic, by having the simulator supply it with coins. The KC after these transformations increases but is still $O(\log n)$, and the simulation is still statistical. Now, to make the simulation perfect, apply Theorem 4.5, and note that by Remark 4.6 the output probability of the simulator, and the determinism of the oracle, are preserved. Finally, we must make the oracle return its answers bit by bit. To do this, we will appropriately encode the answers. Thus, if the response to query z used to be the string a, we now return an appropriate encoding of a: specifically, replace 0 by 00; replace 1 by 11; and terminate with a 01. The (average) KC grows by a constant factor but is still logarithmic. Thus we are in the position assumed above, and now return to it.

Outline. A key step will be the definition of something called a *simulator induced distribution* and some lemmas about it. In particular an important fact will be that the support of the simulator induced distribution Q has density at least $2^{-\kappa}$. Based on Q we will define a PPT machine called a "guessing simulator" M which is a generalization of the simulator in the fraction definition of [GP]— rather than producing the right distribution on a set of density $2^{-\kappa}$, it produces the right distribution when its coins are drawn according to Q. We then follow the paradigm of the proof of [BP, GOP]. Namely, we consider the simulation based prover P^* corresponding to M. We generalize the main lemma of [GOP] to see that (P^*, V) accepts with probability $O(2^{-2\kappa(n)})$ when $x \in L$. We conclude by applying the uniform generation procedure of [BP] to emulate the (P^*, V) system in $\mathsf{BPP}^{\mathsf{NP}}$.

MORE NOTATION. Let $r(n)$ be the number of coins tossed by the simulator S^γ. Since the simulator runs in polynomial time we may fix a polynomial $N = N(n)$ which exceeds the number of bits returned by the oracle. Let $q = r + N$. We are now ready to define the induced distribution and prove the main lemmas about it.

6.1 The simulator induced distribution

We will let $\gamma(R)$ denote the sequence of bits returned by the oracle when the simulator's coins are R, the input x being implicit. We define a distribution Q: $\{0,1\}^q \to [0,1]$ as follows. For $R \in \{0,1\}^r$ and $y \in \{0,1\}^N$ let

$$Q(R\,.\,y) \;=\; \begin{cases} 2^{-r-N+|\gamma(R)|} & \text{if } y \text{ has prefix } \gamma(R) \\ 0 & \text{otherwise.} \end{cases}$$

This Q is called the distribution *induced* by the simulator S^γ.

We will be interested in two kinds of probabilities over $\{0,1\}^q$. We let $\Pr_1[\,\cdot\,]$ denote the uniform distribution over this set, and we let $\Pr_2[\,\cdot\,]$ denote the probability under distribution Q. We let $\mathrm{supp}(Q)$ denote the support of Q, namely the set of all strings appearing with positive probability. A simple observation which will be useful later is:

Lemma 6.1 If $B \subseteq \mathrm{supp}(Q)$ then $\Pr_1[\,B\,] \le \Pr_2[\,B\,]$.

Proof: Just note that $Q(\omega) \ge 2^{-q}$ for all $\omega \in \mathrm{supp}(Q)$. ∎

The key fact about Q is that its support is quite large: it has density at least $2^{-\kappa}$. In other words, the probability of $\mathrm{supp}(Q)$ under the *uniform* distribution is at least $2^{-\kappa}$.

Lemma 6.2 If $x \in L$ then $|\mathrm{supp}(Q)| \cdot 2^{-q} \ge 2^{-\kappa}$.

Proof: We compute:

$$\Pr_1[\mathrm{supp}(Q)\,] \;=\; \mathbf{E}_R\left[\,2^{-|\gamma(R)|}\,\right] \;\ge\; 2^{-\mathbf{E}_R[|\gamma(R)|]}\,,$$

the last step by Jensen's inequality. But $\mathbf{E}_R\,[\,|\gamma(R)|\,]$ is at most $\mathbf{AvCom}_L^{\gamma\to S}(n)$ which by assumption is at most κ. ∎

We now define a machine M_1 (resp. M_2) as follows. On input x it picks a q bit string $\omega = R\,.\,y$ at random (resp. at random according to Q). Now it runs S on coins R, providing oracle answers from the bits of y. (Recall that the oracle returns one bit to each query. Thus the first time a bit is queried, M returns the first bit of y, the second time the second bit of y, and so on, until S stops querying). It outputs whatever S outputs. The machine M_2 has a nice property:

Lemma 6.3 If $x \in L$ then $M_2(x) = \mathrm{View}_V^P(x)$.

Proof: If coins $\omega = R\,.\,y$ are drawn from distribution Q then a particular R appears with probability 2^{-r}, and, given this, y has prefix $\gamma(R)$ with probability 1. Now the lemma follows from the perfectness of the simulator S^γ. ∎

However, M_2 is unlikely to be computable in PPT because it must sample according to Q. However, M_1 is a PPT machine. We call it the *guessing simulator* because it tries to guess oracle answers. But we don't know anything à priori about its behaviour: if, by chance, the first $|\gamma(R)|$ bits of y equal $\gamma(R)$ then the response of M_1 is correct, but otherwise we have no idea, not only what S will do, but even how many bits of y it will use. Nonetheless it is the basis of our efficient proof system.

6.2 The M_1-based proof system

Let $M_1(\omega)$ denote the conversation output by M_1 on coins $\omega \in \{0,1\}^q$. Let Ω_h denote the set of all $\omega \in \{0,1\}^q$ such that $M_1(\omega)$ has prefix h. We let P^* be the M_1-based prover. This is defined like a "simulation based prover," viewing the guessing simulator M_1 as the simulator. Namely, in response to partial conversation h, prover P^* returns m with probability $|\Omega_{hm}|/|\Omega_h|$.

It is important to note that P^*'s moves are based on the actions of M_1 rather than M_2. We know from Lemma 6.3 that M_2 behaves nicely for $x \in L$ and one is tempted to use a simulation based prover derived from M_2. But since we don't know how to compute Q the later step of implementing P^* efficiently would fail. So instead we prove that things work relatively well even under the uniform distribution. Specifically, the following lemma considers the interaction of P^* with the given (honest) verifier V and provides an analogue of [GOP, Lemma 3.1].

Lemma 6.4 If $x \in L$ then the probability that P^* can convince V to accept is at least $(1/4) \cdot 2^{-2\kappa}$.

Note that $(1/4) \cdot 2^{-2\kappa} \geq 1/\operatorname{poly}(n)$ by assumption. On the other hand the soundness of the proof still implies that no P' can convince V to accept when $x \notin L$, except with the negligible probability $\epsilon(n)$. Now, notice that P^*'s computation can be viewed as follows. Pick ω at random from Ω_h and compute the conversation $c = M_1(\omega)$. This has the form $h \,.\, m \,.\, s$ for some m, s. Now return m. Thus the uniform generation procedure of [BP] can be used to implement P^* in probabilistic polynomial time with an NP oracle. (There is some error, but it is exponentially small and can be neglected.) Thus in $\mathsf{BPP}^{\mathsf{NP}}$ we can run the (P^*, V) system on an input x and see whether or not it accepts. This is the final procedure to decide L. It remains to prove Lemma 6.4.

We need some more lemmas. In all of the following it is assumed that $x \in L$. Let A_h denote the set of all $\omega \in \operatorname{supp}(Q)$ such that $M_1(\omega)$ is an accepting conversation prefixed with h. In particular A_λ is the set of all $\omega \in \operatorname{supp}(Q)$ such that $M_1(\omega)$ is an accepting conversation. We begin by showing that this set has reasonably high probability under the *uniform* distribution. (Note this lemma does not directly imply Lemma 6.4. That is a different claim which still needs to be proved!)

Lemma 6.5 $\Pr_1[\, A_\lambda \,] \geq (1/2) \cdot 2^{-\kappa}$.

Proof: We let $S = \operatorname{supp}(Q)$. Since $x \in L$, the probability that a conversation between P and V is accepting is at least $1 - \epsilon(n)$. Now applying Lemma 6.3 we have $\Pr_2[\, A_\lambda \,] \geq 1 - \epsilon(n)$. Thus, $\Pr_2[\, S - A_\lambda \,] \leq \epsilon$. We stress that these facts are under distribution Q, not the uniform distribution. But now we can apply Lemma 6.1 to see that:

$$\Pr_1[\, S - A_\lambda \,] \;\leq\; \Pr_2[\, S - A_\lambda \,] \;\leq\; \epsilon \,.$$

By Lemma 6.2 it follows that $\Pr_1[\, A_\lambda \,] \geq 2^{-\kappa} - \epsilon$. The lemma follows because ϵ is negligible. ∎

Let $\chi(c)$ be 1 if conversation c is accepting and 0 otherwise.

Lemma 6.6 $\chi(c) \geq \Pr_1[\, A_c \,] / \Pr_2[\, \Omega_c \,]$.

Proof: If $\chi(c) = 0$ then $A_c = \emptyset$ so the claim is certainly true. Now suppose $\chi(c) = 1$. Let $S = \operatorname{supp}(Q)$. We have

$$\Pr_1[\, A_c \,] \;\leq\; \Pr_1[\, \Omega_c \cap S \,] \;\leq\; \Pr_2[\, \Omega_c \cap S \,] \;\leq\; \Pr_2[\, \Omega_c \,] \,.$$

The first inequality is because $A_c \subseteq \Omega_c \cap S$ and the second used Lemma 6.1. ∎

We are now ready to begin the proof of Lemma 6.4. We follow and generalize the argument of [GOP, Lemma 3.1]. The idea is to replace their ratios of sets with ratios of probabilities which involve the two *different* distributions, the uniform and Q. The probability that P^* convinces V to

accept on input x is $\mathbf{E}_c[\,\chi(c)\,]$, the expectation being over the distribution on conversations induced by the interaction of P^* and V. By Lemma 6.6 we have:

$$\mathbf{E}_c[\,\chi(c)\,] \;\geq\; \mathbf{E}_c\left[\,\frac{\Pr_1[\,A_c\,]}{\Pr_2[\,\Omega_c\,]}\cdot\frac{\Pr_1[\,A_c\,]}{\Pr_1[\,\Omega_c\,]}\,\right].$$

We will show that

$$\mathbf{E}_c\left[\,\frac{\Pr_1[\,A_c\,]}{\Pr_2[\,\Omega_c\,]}\cdot\frac{\Pr_1[\,A_c\,]}{\Pr_1[\,\Omega_c\,]}\,\right] \;\geq\; \frac{\Pr_1[\,A_\lambda\,]}{\Pr_2[\,\Omega_\lambda\,]}\cdot\frac{\Pr_1[\,A_\lambda\,]}{\Pr_1[\,\Omega_\lambda\,]}\,. \tag{5}$$

But $\Pr_2[\,\Omega_\lambda\,] = \Pr_1[\,\Omega_\lambda\,] = 1$, while $\Pr_1[\,A_\lambda\,] \geq (1/2)\cdot 2^{-\kappa}$ by Lemma 6.5. So showing the above will conclude the proof.

The proof of Equation (5) is by (reverse) induction on the construction of c. We consider separately the case where the current move is by the prover P^*, and the case where the current move is by the verifier V. In both cases we show that for any history h and possible next message m:

$$\mathbf{E}_m\left[\,\frac{\Pr_1[\,A_{hm}\,]}{\Pr_2[\,\Omega_{hm}\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_1[\,\Omega_{hm}\,]}\,\right] \;\geq\; \frac{\Pr_1[\,A_h\,]}{\Pr_2[\,\Omega_h\,]}\cdot\frac{\Pr_1[\,A_h\,]}{\Pr_1[\,\Omega_h\,]}\,. \tag{6}$$

The expectation here is over m chosen according to the $P^* \leftrightarrow V$ interaction given history h. Clearly Equation (5) follows from Equation (6).

We consider first the case of a prover move. Given h we know P^* sends m with probability $\Pr_1[\,\Omega_{hm}\,]/\Pr_1[\,\Omega_h\,]$. Thus:

$$\begin{aligned}
\mathbf{E}_m\left[\,\frac{\Pr_1[\,A_{hm}\,]}{\Pr_2[\,\Omega_{hm}\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_1[\,\Omega_{hm}\,]}\,\right] &= \sum_m \frac{\Pr_1[\,\Omega_{hm}\,]}{\Pr_1[\,\Omega_h\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_2[\,\Omega_{hm}\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_1[\,\Omega_{hm}\,]} \\
&= \frac{1}{\Pr_1[\,\Omega_h\,]}\cdot\sum_m \frac{\Pr_1[\,A_{hm}\,]^2}{\Pr_2[\,\Omega_{hm}\,]} \\
&\geq \frac{1}{\Pr_1[\,\Omega_h\,]}\cdot\frac{\Pr_1[\,A_h\,]^2}{\Pr_2[\,\Omega_h\,]}\,,
\end{aligned}$$

establishing Equation (6) as desired. Here we are exploiting a well known fact, namely that given real numbers $a, b > 0$, the minimum of $\sum_i a_i^2/b_i$, taken over all collections of non-negative real numbers $\{a_i\}, \{b_i\}$ satisfying $a = \sum_i a_i$ and $b = \sum_i b_i$, is a^2/b.

Now we consider the case of a verifier move. Lemma 6.3 implies that given h the verifier sends m with probability $\Pr_2[\,\Omega_{hm}\,]/\Pr_2[\,\Omega_h\,]$. Thus:

$$\begin{aligned}
\mathbf{E}_m\left[\,\frac{\Pr_1[\,A_{hm}\,]}{\Pr_2[\,\Omega_{hm}\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_1[\,\Omega_{hm}\,]}\,\right] &= \sum_m \frac{\Pr_2[\,\Omega_{hm}\,]}{\Pr_2[\,\Omega_h\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_2[\,\Omega_{hm}\,]}\cdot\frac{\Pr_1[\,A_{hm}\,]}{\Pr_1[\,\Omega_{hm}\,]} \\
&= \frac{1}{\Pr_2[\,\Omega_h\,]}\cdot\sum_m \frac{\Pr_1[\,A_{hm}\,]^2}{\Pr_1[\,\Omega_{hm}\,]} \\
&\geq \frac{1}{\Pr_2[\,\Omega_h\,]}\cdot\frac{\Pr_1[\,A_h\,]^2}{\Pr_1[\,\Omega_h\,]}\,,
\end{aligned}$$

establishing Equation (6) as desired, again using the same fact as above. This concludes the proof of Lemma 6.4.

7 Connections with PCP

Oracle entropy enables us also to measure, for example, how much "knowledge" a verifier V obtained in the process of checking a probabilistically checkable proof. Here we would be in the setting of a machine A talking to an oracle described in Section 3, with A being the verifier V and γ the function describing the probabilistically checkable proof (γ takes a query which is a location and returns the bit in the proof string at that location). We observe that the oracle entropy is related not to the amount of communication, as measured by the number of query bits, but can be bounded by the number of *free bits* (cf. [FK, BS]) used in the checking.

In particular, [BS] show that for any NP language L there is a PCP verifier V such that the following is true. V takes input x and a parameter m, and achieves error $O(2^{-m})$ using $3m$ free bits. Furthermore there is an oracle γ such that $\Pr_R[\, V^\gamma(x;R) \text{ accepts}\,] = 1$ for $x \in L$. We claim that $\mathbf{Info}_L^{\gamma\to V}(n) \leq 3m$. Thus, roughly, the verifier gets about 3 bits of knowledge per $1/2$ factor of the error, on inputs in the language.

To prove our claim we need to exhibit an appropriate oracle simulator V' for the [BS] verifier V. On input $x \in L$ and a random string R of V, the oracle simulator will pick $3m$ bits at random and view these as the free bits. By definition of freeness, it can now compute a unique setting for the remaining bits to be queried which would lead V to accept. Thus $P_x(R)$ is at least 2^{-3m} leading to the claimed bound on the oracle entropy.

8 Protocols for set sizes

Background. The theorems of [F, AH] show that $\mathsf{SZK}_{\mathrm{hv}} \subseteq \mathsf{AM}[2] \cap \mathsf{coAM}[2]$. We know that $\mathsf{SKC}^{\mathrm{av}}_{\mathrm{hv}}[O(\log n)] \subseteq \mathsf{BPP}^{\mathsf{NP}}$. Can this be extended to $\mathsf{SKC}^{\mathrm{av}}_{\mathrm{hv}}[O(\log n)] \subseteq \mathsf{AM}[2] \cap \mathsf{coAM}[2]$?

The proofs of [F, AH] involve constant round protocols in which the prover claims and proves bounds on the sizes of certain sets defined by the protocol, with a certain amount of precision. If an analogous approach is to be employed for the knowledge complexity case, the accuracy of the approximate bound must be higher. (This is because the error probability of a $\mathsf{SZK}_{\mathrm{hv}}$ proof can be decreased by parallel repetition while preserving the SZK w.r.t. the honest verifier, but repetition increases the KC.) While constant factor precision sufficed before, one would now want $1 + 1/\operatorname{poly}(n)$ factor precision.

Lower bound protocols of sufficiently high accuracy exist, but the existing upper bound protocol of [F] only provides constant factor accuracy. Here we provide a constant round protocol for upper bounding the size of a set with $1 + 1/\operatorname{poly}(n)$ factor precision.

We stress that we do not know whether proving set sizes with higher accuracy is *sufficient* to prove $\mathsf{SKC}^{\mathrm{av}}_{\mathrm{hv}}[O(\log n)] \subseteq \mathsf{AM}[2] \cap \mathsf{coAM}[2]$. But it might be a useful tool to have.

Setting. The upper and lower bound protocols that we will consider will work for sets generated by functions in FNP. Recall, $f \in \mathsf{FNP}$ if there is a polytime machine, M, such that for all r, there exists a i, $|r| = |i|^{O(1)}$, such that $f(r) = M(r,i)$. Let $\beta(s) = |f^{-1}(s)|$ and $n = |s|$. The common input to both parties is s. In the upper bound protocol the verifier will have an additional private input consisting of a uniformly distributed element of $\beta(s)$.

Lower bound protocols. An (ϵ, q)-lower bound protocol takes as input s, and L, a candidate lower bound for $\beta(s)$. It runs in $O(1)$ rounds and has the following properties: If $L \leq \beta(s)/(1+\epsilon)$ then the verifier accepts with probability at least $1-q$; and If $L > \beta(s)$ then the verifier accepts with probability at most q. For any $\epsilon(n)$ and $q(n)$ satisfying $1/\epsilon(n), \log(1/q(n)) \leq \operatorname{poly}(n)$ there exists a (ϵ, q)-lower bound protocol. (The standard protocol of Babai [B] works for $1/\epsilon(n) = O(1)$.

A protocol for the general case can be derived by applying the Stockmeyer's cross product trick [St]. Alternatively, Babai and Moran provide appropriate protocols more directly [BM].)

UPPER BOUND PROTOCOL. The following is a slight variation on the upper bound protocol of Fortnow [F]. We assume here that all sets have size at least 2^{n^γ} for some $\gamma > 0$. This ensures that there is at least one prime between β and $\beta(1+\epsilon)$ for $1/\epsilon = n^{O(1)}$.

k-Upper Bound Protocol
Input: s, U. In addition the verifier B has a random element of $b \in \beta(s)$ unknown to the prover A.

(1) A sends a proof, w, that U is prime. B rejects if w is not a witness of the primality of U.
(2) B picks h, a random $k+1$-wise independent hash function mod U. B calculates $h(b) = z$ and sends z to A.
(3) A sends c to the verifier and a proof that $c \in \beta(s)$.
(4) B accepts if $b = c$.

Lemma 8.1 Suppose $k \geq 4$. If $U \geq \beta$ then B accepts with probability at least $1 - 1/e - 1.1/k$. If $U \leq \beta(s)(1-\epsilon)$ then B accepts with probability at most $1 - 1/e - (1-2/e)\epsilon + 1.1/k$ for $-1 \leq \epsilon \leq 1$.

The error probability here is a constant. We now provide a protocol which rejects with high probability whenever ϵ is greater than an inverse polynomial. We define a protocol which takes many sets as input but returns just one set with accurate upper and lower bounds if the protocol does not reject.

(k,q)-Tight Bound Protocol
Input: $(s_1, L_1, U_1), \ldots, (s_v, L_v, U_v)$, where $v = k^2 n$. In addition, B has random elements $b_i \in \beta(s_i)$ for all i.

(1) B first checks that $L_i(1+1/k) \leq U_i \leq L_i(1+1/k)^2$ for all i. If not then it rejects.
(2) Next, A and B run $(1/k, 2^{-n})$-lower bound protocols on all (s_i, L_i) in parallel. If one rejects then the verifier rejects.
(3) Next A and B run k-upper bound protocols on all (s_i, L_i) in parallel. Let $\hat{p}$ be the average acceptance rate. If $\hat{p} < 1 - 1/e - 2.1/k$ then the verifier rejects.
(4) Finally, if B has not rejected, it picks a random $I \in \{1, \ldots, v\}$, and output s_I, U_I, L_I.

Let p_i be the probability that the k-upper bound protocol on s_i, U_i accepts. Let $\bar{p}$ be the average of the p_i's.

Lemma 8.2 A can play so that B will accept with overwhelming probability and the output will have $L_i = \beta(s_i)/(1+1/k)$ and $U_i \leq \beta(s_i)(1+1/k)$. If the upper bounds yield $\bar{p} < 1 - 1/e - 3.1/k$ or if $L_i > \beta(s_i)$ for some i then the protocol accepts with probability at most $2^{-\Omega(n)}$. If the upper bounds yield $\bar{p} \geq 1 - 1/e - 3.1/k$, and $L_i \leq \beta(s_i)$ for all i, and the protocol accepts, then there is a constant c such that

$$L_I \frac{(1+1/k)^2}{1 - c/kq} \;\geq\; U_I \frac{1}{1 - c/kq} \;\geq\; \beta(s_I)$$

and

$$\beta(s_I) \;\geq\; L_I \;\geq\; U_I \frac{1}{(1 - c/kq)(1+1/k)^2}\,.$$

8.1 Proof of Lemma 8.1

To calculate the probability that the verifier accepts (i.e., $b = c$), let $\tilde{X}_c$, $c \in \beta - \{b\}$, be the indicator random variable for whether $h(c) = z$. Define $\tilde{X} = \sum_c \tilde{X}_c$ as the total number of elements of β other than b to be mapped to z. The probability that the verifier accepts is just

$$\sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[\tilde{X} = j].$$

Let X_c, $c \in \beta - \{b\}$, be the indicator random variable for whether $H(c) = z$ where H is a random β-wise independent hash function. Of course,

$$\begin{aligned} \sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[\tilde{X} = j] &\leq \Pr[\tilde{X} \leq l-2] + \frac{1}{l} \\ \sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[X = j] &\leq \Pr[X \leq l-2] + \frac{1}{l} \end{aligned}$$

for any l. It can be shown using the Bonferroni inequalities that

$$\left| \Pr[\tilde{X} = j] - \Pr[X = j] \right| \leq \binom{k}{k-j} \binom{\beta-1}{k} \cdot \frac{1}{U^k}$$

for $0 \leq j \leq k-2$. Using these later inequalities, it follows that

$$\begin{aligned} &\left| \sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[\tilde{X} = j] - \sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[X = j] \right| \\ &\leq \frac{1}{k} + \binom{\beta-1}{k} \frac{1}{U^k} \sum_{j=0}^{k-2} \frac{1}{j+1} \binom{k}{j}. \end{aligned}$$

Denote this latter sum by Γ_k. It can shown that

$$\Gamma_k \leq \frac{(\beta-1)_k}{U^k} \frac{2^{k+1}}{(k+1)!} \leq \left(\frac{\beta}{U}\right)^k \left(\frac{2e}{k+1}\right)^k,$$

where the last inequality follows from $n! \geq (n/e)^n / \sqrt{2\pi n}$. Hence, $\Gamma_k \leq 2^{-k}$ for $\beta/U \leq (k+1)/4e$.

If we denote $\sum_{j=0}^{\beta-1} \frac{1}{j+1} \Pr[X = j]$ by $T(U, \beta)$ it can be shown that

$$T(U, \beta) = \frac{U}{\beta} \left(1 - \left(1 - \frac{1}{U}\right)^\beta\right).$$

In particular, for β sufficiently large, we will consider $T(\beta, \beta)$ to be $1 - 1/e$, and ignore the negligible error term. We will need to characterize the behavior of $T(U, \beta)$ as U/β varies. This is done in the following lemma

Lemma 8.3 For β sufficiently big,

- If $U \geq \beta(1+\epsilon)$ then $T(U, \beta) \geq 1 - 1/e + (c/e)\epsilon$, for $\epsilon \leq .1658$, where $c = .359$.
- If $U = \beta(1-\epsilon)$ then $T(U, \beta) \leq 1 - (1/e) - (1 - 2/e)\epsilon$ for all $-1 \leq \epsilon \leq 1$.

The proof of this lemma is omitted. To finish the analysis of the upper bound protocol we must include the difference between $T(U, \beta)$ and the probability that the verifier accepts. Hence, when the prover provides a $U \geq \beta(1+\epsilon)$, then the verifier will accept with probability at least, $1 - 1/e - 1/k - 1/2^k + (c/e)\epsilon$, for $\epsilon \leq .1658$, where $c = .359$.

Suppose the prover selects a $U = \beta(1-\epsilon)$, $-1 \leq \epsilon \leq 1$. If $4e/(k+1) \leq 1 - \epsilon \leq 2$, then the probability that the verifier accepts is at most $1 - 1/e - (1 - 2/e)\epsilon + 1/k + 1/2^k$. To cover the entire range of $1 - \epsilon$ we can use the bound in [AH] who showed that the probability that the verifier accepts when it uses a pair-wise independent hash function is at most $gU/(\beta - 1) \leq g'U/\beta$, where $g = 3 + \sqrt{5}$ and $g' = g + 0.1$. The latter upper bound is bigger than the former for small ϵ. If the crossover point is denoted by ϵ_c, then

$$\epsilon_c = (g' - (1 - 1/e) - 1.1/k)/(g' - (1 - 2/e)).$$

The bound $1-1/e-(1-2/e)\epsilon+1/k+1/2^k$ is indeed valid for $0 \leq \epsilon \leq \epsilon_c$ since $1-\epsilon_c \geq 4e/(k+1)$ for sufficiently large k. If follows that this bound is, in fact, an upper bound for the entire range $-1 \leq \epsilon \leq 1$.

8.2 Proof of Lemma 8.2

The analysis for the prover which plays according to the protocol is simple and is omitted here. The analysis for a cheating prover is as follows. Let p_i be the probability that the k-upper bound protocol on s_i, U_i accepts. Let $\bar{p}$ be the average of the p_i's. Suppose the prover selects upper bounds so that $\bar{p} < 1-1/e-3.1/k$. In such a case $\hat{p} > 1-1/e-2.1/k$ with probability at most $2^{-\alpha n}$ and hence the verifier accepts with probability at most this much. Suppose that the prover selects $L_i > \beta(s_i)$ for some i. Then the verifier accepts the protocol with probability at most 2^{-n}.

Now suppose that the prover selects upper bounds so that $\bar{p} \geq 1-1/e-3.1/k$ and selects $L(s_i) \leq \beta(s_i)$ for all i. This latter condition implies that $U_i \leq \beta(s_i)(1+1/k)^2 \leq \beta(s_i)(1+3/k)$ for all i. This in turn implies that p_i is at most $1-1/e+(1.1+3(1-2/e))/k$ for all i. Now the average probability is very close to the upper bound on all probabilities. This implies that there cannot be too many small probabilities. To quantify this we will use the following standard lemma:

Lemma 8.4 Given n numbers all with value at most M and with average value at least q, the fraction of numbers with value at least $q-\Delta$ is at least $1-\epsilon/\Delta$, where $\epsilon = M-q$.

Using this lemma we see that the fraction of p_i with $p_i \geq 1-1/e-3.1/k-a$ is at least $1-q$ for $a = c'/kq$ for some constant c'. Now, from Lemma 8.1 it follows that $U_i \geq \beta(s_i)(1-c/kq)$ for some constant c.

Acknowledgments

We thank Johan Håstad and Oded Goldreich for valuable comments. In particular we thank the latter for drawing our attention to the implication of our results for the issue of expected versus strict poly-time simulation.

Work done while the first author was at the IBM T.J. Watson Research Center, New York.

References

[AH] W. Aiello and J. Håstad. Perfect Zero-Knowledge can be Recognized in Two Rounds. *J. Computer and System Sciences* **42**, 327–345, 1991.

[B] L. Babai. Trading Group Theory for Randomness. *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing,* ACM, 1985.

[BM] L. Babai and S. Moran. Arthur Merlin Games: a Randomized Proof System and a Hierarchy of Complexity Classes. *JCSS*, Vol. 36, 1988, No. 2, pp 254–276.

[BP] M. Bellare and E. Petrank. Making Zero-Knowledge Provers Efficient. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[BS] M. Bellare and M. Sudan. Improved non-approximability results. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[DDPY] A. De Santis, G. Di Crescenzo, G. Persiano and M. Yung. On monotone formula closure of SZK. *Proceedings of the Thirty Fifth Annual Symposium on the Foundations of Computer Science*, IEEE, 1994.

[FK] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994.

[F] L. Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research* Vol. 18, Ed. S. Micali, 1989.

[GP] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science*, IEEE, 1991.

[GOP] O. Goldreich, R. Ostrovsky and E. Petrank. Computational complexity and knowledge complexity. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994.

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

[M] H. Maier, Primes in Short Intervals, Michigan Math, **J 32**, 1985, pp:221-225.

[St] L. Stockmeyer. The Complexity of Approximate Counting. *Proceedings of the Fifteenth Annual Symposium on the Theory of Computing*, ACM, 1983.

A Probabilistic oracles

We extend the definition of oracle entropy given in Section 3 to the setting where the oracle supplied to the simulator is probabilistic. We should note, however, that this is relevant only when one cannot neglect even small errors. Otherwise, a probabilistic oracle may be made determinstic by having its caller supply it with coins, at cost of a negligible statistical variation in the output of A.

Again we are in the setting of a machine A querying oracle γ. The oracle's being probabilistic means that its response to any query q is computed by picking $\rho \in \{0,1\}^\infty$ at random and returning $\gamma(q;\rho)$. Thus $A^\gamma(x;R)$ is now a random variable for each x, R. The definition below is inspired by the "fraction" definition of KC of [GP], but different in some ways. Let A' be a PPT machine, tossing $r'(\cdot)$ coins. Require that for all $x \in L$ and all $R \in \{0,1\}^{r(n)}$ there is a "good" subset $G_x(R) \subseteq \{0,1\}^{r'(n)}$ satisfying ${A'}_{G_x(R)}(x,R) = A^\gamma(x;R)$, where ${A'}_{G_x(R)}(x,R)$ is the random variable whose output is computed by picking ρ at random from $G_{x,R}$ and returning $A'(x,R;\rho)$. (That is, it is A' with its coin tosses chosen randomly from the subset $G_{x,R}$ of all the possible coin tosses.) Now let $\mathbf{Info}_{A'}^{\gamma\to A}(x)$ equal

$$\mathbf{E}_{R \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{r(n)}} \left[\lg \frac{1}{\Pr_{\rho \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{r'(n)}} [\, \rho \in G_x(R) \,]} \right].$$

We say that $\mathbf{Info}_L^{\gamma\to A}(n) \le \kappa(n)$ if there is an oracle simulator A' as above such that $\mathbf{Info}_{A'}^{\gamma\to A}(x) \le \kappa(|x|)$ for all $x \in L$. We leave to the reader to check that when the oracle γ is deterministic the definition above coincides with that in Section 2.