Relativized Perfect Zero Knowledge is not BPP.

William Aiello* Johan Hastad**

Applied Math Department and Laboratory of

Computer Science, MIT

Abstract: In this paper we further study the complexity of of zero-knowledge interactive proofs. We prove that there is an oracle A such that there is a language L which is recognizable by a two round, perfect zero-knowledge interactive proof relative to A, but such that $L \notin BPP^A$. This gives interesting implications for what can be demonstrated about zero-knowledge interactive proofs using standard methods.

1. Introduction

Interactive proofs were independently introduced by Babai [B] and Goldwasser, Micali, and Rackoff [GMR], The class IP is defined through the computational model of an interactive prover-verifier pair. Both Turing machines in a pair receive a common input, w, and exchange up to a polynomial in $|w|$ number of messages, each of length a polynomial in $|w|$. The verifier's moves and its final determination of whether to accept or reject w are the result of random polynomial time computations on w and all messages sent so far. The prover has no resource bounds. A language, L, is in $IP[f(n)]$ if there exists an

* Supported by an ONR fellowship and partially supported by NSF grant DCR-8509905.

** Supported by an IBM Post Doctoral Fellowship and partially supported by Air Force Contract AFOSR-86-0078. Currently at the Royal Institute of Technology.

interactive prover-verifier pair that on input w exchanges at most $f(|w|)$ messages such that: 1.) when $w \in L$, the verifier interacting with the prover accepts with probability at least $1 - 2^{-|w|}$ and, 2.) when $w \notin L$, the verifier interacting with any prover accepts with probability at most $2^{-|w|}$. Such a prover-verifier pair is called an interactive proof for L. Let $IP = \cup_k IP[n^k]$. Just as in the case when $L \in NP$, when $L \in IP$, membership in L is efficiently verifiable since the verifier runs in polynomial time and determines membership correctly with probability very close to one. However, IP is thought to strictly contain NP since it has recently been shown by Shamir [S] that $IP = PSPACE$.

In addition to defining interactive proofs, Goldwasser, Micali, and Rackoff [GMR] further defined zero-knowledge interactive proofs. The zero-knowledge definition was motivated by cryptographic considerations (see for example, [GMR2], [O], [GMW]). Informally, a prover is zero-knowledge for a language if the prover reveals no useful information (other than language membership) when interacting with any verifier. Slightly more formally, a prover is zero-knowledge for L if for any verifier there is a probabilistic polynomial time simulator that, on inputs in L, produces conversations with the "same" probability distribution as the prover interacting with that verifier. Actually, three interpretations of "same" lead to three types of zero-knowledge, each more restrictive than the next. When "same" is informally interpreted as: 1.) identical, 2.) almost identical, or 3.) equivalent with respect to probabilistic polynomial time, then the prover is said to be perfect zero-knowledge, statistical zero-knowledge, or computational zero-knowledge for L, respectively. A language, L, is in PZK (SZK, CZK) if there is an interacting prover-verifier pair which is an interactive proof for L with the additional property that the prover is perfect (statistical, computational) zero-knowledge for L.

In this paper we continue the investigation of the complexity-theoretic implications of the zero-knowledge definitions. Requiring that, for inputs in the language, the conversations between the prover and every verifier be accurately reproducible by some random polynomial time machine would seem to be a severe constraint on the power of the prover and hence the power of the zero-knowledge model. Surprisingly, for computational zero-knowledge this is probably not the case. Through the work of [GMW], [BGGHKMR], [IY] it has been shown that, assuming secure encryption exists, any interactive proof can be transformed into a computational zero-knowledge proof, i.e., $CZK = IP$.

However, the inuition that zero-knowledge is very restrictive seems to be correct for statistical and perfect zero-knowledge. Fortnow [F] was the first to provide evidence that the statistical zero-knowledge requirement may restrict the power of the prover. He proved that if a language has a statistical zero-knowledge proof, then the complement of the language has a bounded round interactive proof, i.e., $SZK \subseteq \textit{co-}IP[2]$. From this theorem we can deduce that it is unlikely that SZK contains all of NP since if $NP \subseteq SZK$ then $\textit{co-}NP \subseteq IP[2]$, which further implies that the polynomial time hierarchy collapses to $IP[2]$ by [BHZ].

While Fortnow's result did imply that SZK is probably weaker than IP it still left open the possibility that SZK contained languages in IP which required a polynomial number of interactions. Aiello and Hastad [AH] show that this cannot be the case. They prove that any language which is recognized by an unbounded round statistical zero-knowledge proof can also be recognized by a two round interactive proof, i.e., $SZK \subseteq IP[2]$. We should note that that $IP[2] \subset \Pi_2$ [B]. Hence, under the assumption that $\Pi_2 \neq PSPACE$, $SZK \neq IP$, that is, perfect or statistical zero-knowledge is a strong constraint on the power of interactive proofs with a polynomial number of rounds. Given this, a natural question to investigate is, "How weak are PZK and SZK?"

There is strong evidence that $PZK \neq BPP$ since Graph Isomorphism (GI), and Quadratic Residuosity (QR) are in PZK but have resisted all attempts to be place in BPP. Furthermore, it is probably also true that $PZK[4] \neq BPP$ since Quadratic Non-Residuosity (QNR) and Graph Non-Isomorphism (GNI) are known to be in $PZK[4]$ but not in BPP.

However, there are no known candidate languages in $SZK[3] - BPP$. In this paper we look for evidence that $SZK[3]$ and languages lower in the zero-knowledge hierarchy are not trivially BPP. Our two main results are two oracles A and B such that $SZK^A[2] \neq BPP^A$ and $PZK^B[2] \neq BPP^B$ respectively. The proofs require diagonalization arguments involving four types of computing devices: a prover, many verifiers, a simulator for each verifier, and all BPP machines. It is the only known oracle separation involving a zero-knowledge complexity class.

The first result is weaker than the second but the former oracle seems to have the interesting property that $SZK \neq PZK$. Even though we cannot prove this last property there is still an interesting observation to be made. All known statistical zero-knowledge proofs have been converted to perfect zero-knowledge proofs by letting the simulator run for a long time with exponentially small probability. This procedure is not possible for our language.

Finally, our results in conjuction with a result of Oren [O] give evidence that the original definition of zero-knowledge proposed in [GMR] is in fact less restrictive then the auxilary input model proposed in sereral papers [O], [TW], [GMR2]. Oren showed that in the auxillary input model $CZK[2] = BPP$. Our results hold in a model which is only slightly less restrictive than the auxiliary input model (and more restrictive than the original definition). We can conclude that any proof of equivalence between the various models cannot relativize.

The content of the paper is organized as follows. In section 2 we give the necessary definitions and notation. In section 3 we construct an oracle such that $SZK^A[2] \neq BPP^A$ and in section 4 we show how to modify this construction to make the protocol perfect zero-knowledge.

2. Notation and Definitions

In this section we give the formal definitions needed for the paper. Let P denote a prover: any probabilistic Turing machine which has a "communication" tape (for a formal definition of a "communication" tape see [GMR]). P has no resource bounds. Let V denote a verifier: any probabilistic polynomial time Turing machine with a communication tape. Let $P \leftrightarrow V$ denote an interacting prover-verifier pair: any prover and verifier which share the same input tape and communication tape (initially empty) and interact in rounds in the following way.

(1) The verifier, V, makes a probabilistic polynomial time computation based on the input, the contents of its memory, and all messages thus far received over the communication tape from the prover, P.

(2) V transmits the result of the computation over the communication tape to P. We will denote the message sent by V in round i by x_{2i-1}.

(3) P performs a probabilistic computation based on the input, and all messages thus far received over the communication tape from V.

(4) P transmits the result of the computation over the communication tape to V. We will denote the message sent by P in round i by y_{2i}.

The interaction is terminated by the verifier accepting or rejecting after at most a polynomial (in the input length) number of rounds.

Let $P \leftrightarrow V(w)$ denote a transcript of the interaction between the prover and the verifier. This is of course a stochastic variable depending on P's and V's random choices.

Definition: A given $P \leftrightarrow V$ is *α-complete* for a language, L, if for all $w \in L$ the probability that V accepts on w is at least α.

Definition: A verifier, V, is *β-sound* for a language, L, if for all $P' \leftrightarrow V$ and all $w \notin L$ the probability that V rejects on w is at least β.

[GMR1] defined the class IP as follows. L is in $IP[f(n)]$ if there exists an interacting prover-verifier pair, $P \leftrightarrow V$, that exchanges at most $f(n)$ messages (n being the length of the input) such that:

1.) $P \leftrightarrow V$ is $(1 - 2^{-n})$-complete for L, and
2.) V is $(1 - 2^{-n})$-sound for L.

Call such a $P \leftrightarrow V$ an interactive proof for L. Note that membership in L is still efficiently verifiable since V runs in polynomial time and verifies membership correctly with probability very close to one. Define IP as the union over k of $IP[n^k]$.

2.1 Zero-Knowledge

In this section we will give the formal definition of a zero-knowledge interactive proof for a language. We will first need some properties of probability distributions on strings.

Let $A(w)$ and $B(w)$ be two parameterized discrete random variables. Let $A[L] = \{A(w) \mid w \in L\}$ and similarly define $B[L]$. We say that $A[L]$ is degree d bounded if

$$\exists d \, \forall w \in L \, \forall y \in A(w), \qquad |y| = |w|^d.$$

We define three types of equivalence between $A[L]$ and $B[L]$.

1.) $A[L] \equiv_P B[L]$ or $A[L]$ is *perfectly* equivalent to $B[L]$ if for all y, and all $w \in L$,

$$Pr[A(w) = y] \; = \; Pr[B(w) = y].$$

2.) $A[L] \equiv_{S(k,N)} B[L]$ or $A[L]$ is (k, N)-*statistically* equivalent to $B[L]$ if for all $w \in L$, $|w| \geq N$,

$$\sum_y |Pr[A(w) = y] - Pr[B(w) = y]| \leq \frac{1}{|w|^k}.$$

3.) $A[L] \equiv_{C(k,N)} B[L]$ or $A[L]$ is (k, N)-*computationally* equivalent to $B[L]$ if whenever $A[L]$ and $B[L]$ are degree d bounded then for all circuit families, C, of size $|w|^k$, and all $w \in L$, $|w| \geq N$,

$$|\; \Pr[C_{|w|^d}(A(w)) = 1] - \Pr[C_{|w|^d}(B(w)) = 1] \;| \leq \frac{1}{|w|^k}$$

where C_n is the circuit of C which takes inputs of size n.

We have already seen what it means for an interacting prover-verifier pair to be an interactive proof for a language. In a cryptographic setting, however, we may require more from our protocol than just completeness and soundness. We may want the prover to give nothing to any verifier (even those not following the protocol) that the verifier could not have computed itself. To formalize this [GMR1] introduced the important concept of a simulator. A simulator, M, is a random Turing machine that produces strings, i.e., "conversations," in expected polynomial time.

Let $M(w)$ be the random variable associated with M on input w. Recall that $P\leftrightarrow V(w)$ is the random variable associated with the conversations produced by $P\leftrightarrow V$ on input w. We will say that P is *statistical* zero-knowledge for L if $\forall k\ \exists N$ such that $\forall V'\ \exists M_{V'}$ such that

$$M_{V'}[L] \equiv_{S(k,N)} P\leftrightarrow V'[L].$$

Define the class SZK to be those languages, L, for which there exists an interactive prover-verifier pair, $P\leftrightarrow V$, such that:

1.) $P\leftrightarrow V$ is $(1-2^{-n})$-complete on L,

2.) V is $(1-2^{-n})$-sound on L, and

3.) P is statistical zero-knowledge for L.

Call such a $P\leftrightarrow V$ a statistical zero-knowledge proof for L.

By using the definitions for perfect equivalence and computational equivalence from above we get similar definitions for *perfect* zero-knowledge (SZK), and *computational* zero-knowledge (CZK).

Several papers have noted that the above definitions may not be restrictive enough. For example, the prover cannot be sure that the verifier's worktapes are empty when both parties receive the input. For example, the worktapes may not have been cleared after the verifier completed a previous interaction with another prover. This was noted in [GMR2], [O], and [TW]. The following definitions handle these cases.

Let $P \leftrightarrow V(w, u)$ denote the random variable for the output of the protocol on input w when V runs in polynomial time in $|w|$ but has additional input u that is unknown to P . Let $M(w, u)$ be the random variable for the output of the simulator on input w, u where the simulator runs in polynomial time in $|w|$. P is *statistical* zero-knowledge$'$ on L if $\forall k$ $\exists N$ such that $\forall V'$ $\exists M_{V'}$ such that

$$M_{V'}[L \times \Sigma^*] \equiv_{S(k,N)} P \leftrightarrow V'[L \times \Sigma^*].$$

The requirements for P to be *perfect* zero-knowledge$'$ on L and *computational* zero-knowledge$'$ on L are similar.

Now we can define the class SZK'. The classes PZK' and CZK' are defined similarly. A language, L, is in SZK' if there exists an interactive prover-verifier pair such that:

1.) $P \leftrightarrow V$ is $(1 - 2^{-n})$-complete on L,

2.) V is $(1 - 2^{-n})$-sound on L, and

3.) P is statistical zero-knowledge$'$ on L.

Call such a $P \leftrightarrow V$ a statistical zero-knowledge$'$ proof for L.

The zero-knowledge$'$ definitions were designed so that zero-knowledge$'$ proofs would be modular. Tompa and Woll [TW] demonstated that if a zero-knowledge$'$ protocol is iterated sequentially then the resulting protocol is zero-knowledge$'$. Oren [O] showed more generally that if a protocol is composed only of zero-knowledge$'$ subprotocols then the protocol itself is zero-knowledge$'$.

In this paper we study relativized zero-knowledge proofs. In this case both the prover and the verifier have access to an oracle A. As far as we know this is the first attempt to study relativized zero-knowledge protocols.

3. An oracle A such that $SZK^A[2] \neq BPP^A$.

In this section we will prove the following theorem.

Theorem 1: *There exists an oracle, A, such that $SZK^A[2] \neq BPP^A$.*

As is standard for relativized separation arguments, we first define a map from an arbitrary set of strings, $A \subset \Sigma^*$, to a set of unary strings, L_A. We then proceed to show that there exists an A such that L_A is in $SZK^A[2]$ but not in BPP^A. Let a^n be the characteristic vector of $A \cap \Sigma^n$. That is,

$$a_i^n = 1 \Longleftrightarrow i \in A \cap \Sigma^n.$$

Divide the first $\lfloor 2^n/3n \rfloor$ of a^n into segments of length $3n$. We will ignore the remaining portions of a^n. Let s_j denote the jth segment, i.e., the positions $1 + 3n(j-1)$ to $3nj$. For each string, v, of length $3n$ define R_v as the segments which have value v: $R = \{i \mid s_i = v\}$. We will say a^n is *unique* whenever $|R_v| \leq 1$ for all strings v of length $3n$. Below we will also need the following definitions. Call a^n *redundant* whenever there are exactly $\lfloor \sqrt{2^n/3n} \rfloor$ v with $\lfloor \sqrt{2^n/3n} \rfloor \leq |R_v| \leq \lfloor \sqrt{2^n/3n} \rfloor + 2$ and $|R_v| = 0$ for the remaining v. Call a^n *completely redundant* if it is the zero vector of length 2^n.

We define the unary language L_A as follows:

$$1^n \in L_A \Longleftrightarrow a^n \text{ is unique.}$$

We will show that there exists an oracle, A, such that $L_A \in SZK^A[2]$ but $L_A \notin BPP^A$. As a first step let us show that there are many oracles for which $L_A \in IP^A[2]$.

Lemma 2: *If A is such that for all n either a^n is unique, redundant, or completely redundant then $L_A \in IP^A[2]$.*

Proof: Consider the following interactive protocol. On input 1^n:

1. V uniformly picks a number j between 1 and $\lfloor 2^n/3n \rfloor$. V asks $3n$ oracle queries to discover segment s_j of a^n. Call this $3n$ bit string c. V sends c to P.
2. If none of the segments of a^n are the string c then P responds "you cheat". If c occurs as segment(s) $s_{i_1}, s_{i_2}, \ldots, s_{i_l}$, P returns $k \in \{i_1, i_2, \ldots, i_l\}$.
3. If $k = j$ then the verifier accepts, otherwise it rejects.

We have to prove that P can win the game with high probability precisely when $1^n \in L_A$. If $1^n \in L_A$ then by definition a^n is unique and the prover will know that c is the jth segment of a^n. Hence, the verifier will accept with probability one. If $1^n \notin L_A$ then by hypothesis a^n is redundant or completely redundant. Hence, there are at least $\lfloor \sqrt{2^n/3n} \rfloor$ segments which are the same as c. So, the prover will only be able to guess j of the verifier with probability at most $\lfloor \sqrt{2^n/3n} \rfloor^{-1}$.

$\square$

Henceforth, we will restrict our attention to oracles A such that a^n is either unique, redundant, or completely redundant. We have to construct A to simultaneously acheive $L_A \in SZK^A[2]$ and $L \notin BPP^A$. We will first treat these two conditions separately and then show how to combine the two requirements.

3.1 Diagonalization over BPP

We will use the standard technique of diagonalization. We will need an enumeration of oracle-BPP machines: M_1^A, M_2^A,.... We will set A in rounds so that at round i, M_i^A will not BPP-recognize L_A. We will say that a machine strong BPP-recognizes a language if it erroneously accepts or rejects with probability at most 2^{-n}. The following lemma will establish that it is sufficient to set A in rounds so that at round i, M_i^A will not strong BPP-recognize L_A.

Lemma 3: *Any language, L, in BPP^A can be recognized by a machine, M^A, which erroneously accepts or rejects with probability at most 2^{-n} where n is the length of the input.*

Proof: As is standard, M^A simulates the machine recognizing L a polynomial number of times and accepts if a majority of the simulations accept.

$\square$

Now back to the diagonalization. We assume without loss of generality that M_i^A runs in time at most n^i on inputs of length n. Will will determine A in rounds by putting strings in and out of the oracle set. A string which has not yet been put in or out of A will be called undetermined.

The general idea of the construction is to ensure that at round i either $1^{n_i} \in L_A$ but $\Pr[M_i^A \textit{ accepts } 1^{n_i}] < 1 - 2^{-n_i}$ or $1^{n_i} \notin L_A$ but $\Pr[M_i^A \textit{accepts } 1^{n_i}] > 2^{-n_i}$. In both cases we have ensured that M_i^A does not strong BPP-recognize L_A .

Let m_i for $i = 1, 2, \ldots$ be defined by $\min_{m \in \mathbf{N}}(m^{2i+1/2}2^{-m/2} \leq 1/2)$. Let $n_1, n_2, \ldots$ be a sequence of intergers defined by $n_1 = \max\{20, m_1\}$ and $n_i = \max(n_{i-1}^{i-1} + 1, m_i)$ for $i = 2, 3, \ldots$. For all x not of length n_i for some i set $x \notin A$. Now we will set the strings of length n_i in rounds.

Round i: Run M_i^A on input 1^{n_i}. Note that since M_i can run for time at most n_i^i it cannot ask about strings of length n_{i+1} or greater. So when M_i^A asks A about a string y we have 3 cases: $|y| = n_j$ for $j < i$, $|y| \neq n_j$ for $1 \leq j \leq i+1$, and $|y_i| = n_i$. In the first case the answer has already been determined in a previous round and in the second case the answer was determined ahead of time. Thus the probability that M_i^A accepts is only a function of the answers to questions of the third type, i.e., of a^{n_i}. Note that no strings of length n_i have been set in previus rounds since n_i is greater than the running time of M_k on 1^{n_k} for $k < i$. Let $p(a^{n_i}) = \Pr_r[M_i^A \textit{ accepts } 1^{n_i}]$ where r is the random coins used by M_i^A. We have two case

1. There exists a redundant a^{n_i} such that $p(a^{n_i}) > 2^{-n_i}$. In this case we set A according to this a^{n_i}.
2. If no redundant a^{n_i} exists with $p(a^{n_i}) > 2^{-n_i}$ find a unique a^{n_i} with $p(a^{n_i}) \leq \frac{1}{2}$ and set A accordingly.

Clearly if this construction is possible we can make any BPP machine make an error. We need only check that part 2 of the construction is possible. Lemma 4 will establish that this is indeed the case. Let U be the set of all a^{n_i} which are unique and R be the set of all a^{n_i} which are redundant.

Lemma 4: *If $p(a^{n_i}) \leq 2^{-n_i}$ for all $a^{n_i} \in R$ then the fraction of U with $p(a^{n_i}) < 1/2$ is $\geq 1 - 2^{2-n_i}$.*

Let us start by giving the intuition behind the proof. Oracle BPP machines behave similarly on both unique and redundant oracle vectors. In essence this is due to the fact that it is difficult for oracle BPP machines to distinguish between unique and redundant oracle vectors. In a redundant vector there are exponentially many different values in the segments. It is very unlikely that a BPP machine sampling a polynomial number of segments will find a pair of segments with identitical value. To make this formal we will fix the random coins of M_i^A and now look at the probability that M_i^A accepts as a function of random a^{n_i}. We have

Lemma 5: *For fixed random coins r,*

$$\Pr_{a^{n_i} \in U}[M_i^A \text{ accepts}] \leq \frac{\Pr_{a^{n_i} \in R}[M_i^A \text{ accepts}]}{1 - n_i^{2i+1/2} 2^{-n_i/2}}$$

Proof: Once we fix the random coins of M_i^A it becomes deterministic and its computation depends only on a^{n_i}. Consider an accepting computation during which M_i^A examines k segments. Let m^R be the number of redundant a^{n_i} on which M_i^A would produce this computation (i.e., which have the same values at those k segments) and let m^U be the number of unique a^{n_i} on which M_i^A would produce this same computation. Assume for now that the values of these k segments are unique. We have

$$\frac{m^U}{|U|} = \prod_{i=0}^{k-1} (2^{3n} - i)^{-1}$$

and

$$\frac{m^R}{|R|} = \prod_{i=0}^{k-1} (2^{3n} - i)^{-1} \cdot \Pr_{a^{n_i} \in R}[k \text{ specified segments have unique values}]$$

We have to estimate the last probability. The probability that any two segments are equal is

$$\leq \frac{\lfloor \sqrt{2^n/3n} \rfloor + 1}{\lfloor 2^n/3n \rfloor} \leq \frac{\sqrt{2^n/3n} + 1}{2^n/3n - 1} = \frac{1}{\sqrt{\frac{2^n}{3n}} - 1} \leq \sqrt{\frac{4n}{2^n}}$$

for $n \geq 20$. Since we have $\binom{k}{2} \leq k^2/2$ different pairs that could be equal we have $\Pr_{a^{n_i} \in R}[$ k chosen segments are unique$] \geq 1 - k^2 \sqrt{n_i} 2^{-n_i/2}$. Since $k \leq n^i$ we have

$$\frac{m^R}{|R|} \geq \frac{m^U}{|U|} \left(1 - n_i^{2i+1/2} 2^{-n_i/2}\right).$$

The inequality is obvious in the case when the values of the k segments are not all unique. Now Lemma 5 follows by summing over all possible accepting computations.

□

Let us now prove Lemma 4. By the hypothesis $p(a^{n_i}) \leq 2^{-n_i}$ for all $a^{n_i} \in R$. It follows that

$$\Pr_{r, a^{n_i} \in R}[M_i^A \text{ accepts } 1^{n_i}] < 2^{-n_i}$$

By Lemma 5 and the choice of n_i

$$\Pr_{r, a^{n_i} \in U}[M_i^A \text{accepts } 1^{n_i}] \leq 2 \Pr_{r, a^{n_i} \in R}[M_i^A \text{ accepts } 1^{n_i}] < 2^{1-n_i}$$

which in its turn implies

$$\Pr_{a^{n_i} \in U}\left[p(a^{n_i}) \geq \frac{1}{2}\right] = \Pr_{a^{n_i} \in U}\left[\Pr_r[M_i^A \textit{ accepts } 1^{n_i}] \geq \frac{1}{2}\right] \leq 2^{2-n_i}.$$

□

3.2 Statistical Zero Knowledge

We have seen how to diagonalize over all oracle BPP machines. Now we will show that for many A the two round protocol for recognizing L_A described earlier is in fact a statistical zero knowledge protocol.

We will show that for any verifier, V', there is a polynomial time coin flipping simulator, $M_{V'}^A$, which for all $1^n \in L_A$ produces conversations with close to the same distribution as the $P^A \leftrightarrow V'^A$ protocol. To make precise what is meant by close define $S_{V'}(a^n)$ as the difference between the distributions of conversations on input 1^n.

$$S_{V'}(a^n) = \sum_{x,y} \left|\Pr[P^A \leftrightarrow V'^A(1^n) = x, y] - \Pr[M_{V'}^A(1^n) = x, y]\right|$$

where x denotes the verifier's move and y denotes the prover's move. Strictly speaking the value of $S_{V'}(a^n)$ does not only depend on a^n since V' might ask questions of lengths other than n. However, we assume any fixed set of answers for stings shorter than n and that any string the verifier asks for of length greater than n gets a negative answer. Observe that this agrees with the situation in the diagonalization over BPP-machines.

To prove that the protocol is SZK we have to prove that $S_{V'}(a^n) < 1/n^c$ for all c and sufficiently large n.

The intuition for the proof is quite clear. If we are dealing with an honest verifier which behaves according to the protocol then for unique a^n the verifier does not learn anything when the prover reveals the location of the segment that the verifier itself chose. It is easy to simulate the honest verifier. The simulator simply runs V to get j and $c = s_j$ and reports that the prover responds with j. On the other hand, consider a verifier which produces c without looking at the oracle. In this case the verifier could learn the position of c in a^n from the prover even though it would be difficult for the verifier to discover this for itself. However, with overwhelming probability c will not be a segment in a^n since there are 2^{3n} strings of length $3n$ but only $\lfloor 2^n/3n \rfloor$ segments of a^n. Thus the simulator could report "you cheat" as the prover's mover and be correct almost always. Let us describe the simulator in more detail.

Definition of simulator : On input 1^n, the simulator runs V'^A which produces a string c. Next the simulator queries the oracle to obtain all segments of a^n in which V'^A has asked at least one question. If one of these segments is equal to c, $M_{V'}$ returns its index as the provers answer and otherwise it returns "you cheat".

Given this definition of $M^A_{V'}$ let's evaluate $S_{V'}(a^n)$. Note that V'^A and $M^A_{V'}$ produce c with the same distribution. If c is not one of the segments of a^n then both the real prover and the simulator prover respond "you cheat". So, the sum over those c which are not segments of a^n contributes zero to $S_{V'}(a^n)$. Say c is the jth segment of a^n and the jth segment was examined by V'^A during the production of c. This happens with the same probability for both ${V'}^A$ and $M_{V'}^A$ and in both cases the prover's response is j. So again, the sum over such c's contributes zero to $S_{V'}(a^n)$. Say, c is the jth segment of a^n but the jth segment was not examined during the production of c. Again this occurs with the same probability for both ${V'}^A$ and $M_{V'}^A$ but in the former case the real prover will respond with j and the simulator prover will respond "you cheat". This is the only case the simulator will not be able to simulate.

Let $D(r, a^n, c)$ be the event that the verifier with coins r and oracle vector a^n produces c which is a segment of a^n but the verifier makes no queries to that segment. By the above argument $S_{V'}(a^n)$ reduces to

$$S_{V'}(a^n) = 2\sum_b \Pr_r[D(r, a^n, b)].$$

Lemma 6: For any V' and any n

$$\Pr_{a^n \in U}[S_{V'}(a^n) > \frac{2}{2^n 3n}] < \frac{1}{2^n}$$

Proof: We'll show that the expected value of $S_{V'}(a^n)$ taken over $a^n \in U$ is at most $2/2^{2n}3n$ and this implies the lemma. The expectation of $S_{V'}(a^n)$ over $a^n \in U$ is

$$\frac{2}{|U|}\sum_{a^n}\sum_b \Pr_r[D(r, a^n, b)]$$

Treat the probability over r as a sum over r weighted by 2^{-p} where p is the number of coins used by the verifier. Change the order of summation to sum over r first and c last. Then note that once r and a^n are fixed c is determined. So ignore the sum over c. We get

$$\mathrm{E}\left(S_{V'}(a^n)\right) = \frac{2}{2^p}\sum_r \Pr_{a^n \in U}[D(r, a^n)]$$

Fix r, clearly if the machine looks at k segments and then produces a string c which is not equal to one of these segments then the probability that c is a segment of a random unique a^n is $\frac{\lfloor \frac{2^n}{3n} \rfloor - k}{2^{3n} - k} \leq (3n2^{2n})^{-1}$. Thus $E(S_{V'}(a^n)) \leq 2(3n2^{2n})^{-1}$ and the lemma follows.

□

3.3 Interleaving the two conditions

Let us see how to choose our oracle to ensure that our language is statistical zero-knowledge while diagonalizing over oracle BPP machines. We only need a slight modification to the procedure in section 3.1: let $V_1, V_2, \ldots$ be an enumeration of probabilistic polynomial time turing machines and replace condition 2 by $2'$.

$2'$. If no redundant a^{n_i} exists with $p(a^{n_i}) > 2^{-n_i}$ find a unique a^{n_i} with $p(a^{n_i}) \leq \frac{1}{2}$ and such that $S_{V_j}(a^{n_i}) \leq \frac{2}{2^{n_i} 3n_i}$ for $j = 1, 2 \ldots i$. Set A according to this a^{n_i}.

Observe that by Lemmas 4 and 6 a random a^{n_i} satisfies the two conditions with probability $\geq 1 - (4+i)2^{-n_i} > 0$ and thus there is such an a^{n_i}.

Finally, to conclude that the construction is correct we need to verify the following conditions.

1. A is well defined.
2. For any n, a^n is unique, redundant or completely redundant.
3. No BPP^A machine recognizes L_A.
4. Every V_i can be simulated.

Let us verify these conditions one at the time.

(1) We need only observe that no strings of length $\geq n_i$ are set under the first $i-1$ rounds. This is true since $n_i > n_k^k$ for $k < i$ and thus no M_k^A can ask about any string of length n_i during its computation.

(2) If $n = n_i$ for some i, a^n is either redundant or unique according to the rules 1 and 2 in the construction. For all other n, a^n is completely redundant.

(3) By the construction M_i^A makes an error on input 1^{n_i}.

(4) Observe that this is only a condition when $1^n \in L$. We know by condition $2'$ in the construction that the distance between the distribution generated by P and V_i on input 1^n and the distribution generated by M_{V_i} on 1^n is $\leq \frac{2}{2^n 3n}$ for $n \geq n_i$.

This finishes the proof of Theorem 1.

$\square$

4. An oracle such that $PZK[2]^B \neq BPP^B$.

Using an idea of Oded Goldreich we show how to modify the language and the oracle to prove the following theorem.

Theorem 2: *There exists an oracle, B, such that $PZK^B[2] \not\subseteq BPP^B$.*

We will define a new language L'_B which is very similar to the previous language. This time we let b^n denote the characteristic vector of elements of B with leading 0 and length $n+1$. That is,

$$b_i^n = 1 \Longleftrightarrow 0i \in B \cap \Sigma^{n+1}.$$

Now we define a unary language L'_B by

$$b^n \textit{ unique} \Longleftrightarrow 1^n \in L'_B.$$

Again we will make sure that b^n is either unique, redundant, or completely redundant.

The key difference is that we will use strings in the oracle starting with 1 as a dictionary. We let $1x \in B$ iff $|x| = 3n$ and x is a segment of b^n. We will call the characteristic vector of this "dictionary" part d^n: for $|i| = 3n$

$$d_i^n = 1 \Longleftrightarrow 1i \in B \Longleftrightarrow i \text{ is a segment of } b^n.$$

The proof of Theorem 2 will be very similar to the proof of Theorem 1 and hence we will omit some details.

4.1 Diagonalization over BPP

As before we start with an enumeration of oracle-BPP machines: M_1^B, M_2^B,.... Let m_i and n_i be defined as before. For all x not of length n_i for some i set $0x \notin B$. Also for all x not of length $3n_i$ for some i set $1x \notin B$. Again we will determine the remainder of the oracle in rounds.

Round i: Run M_i^B on input 1^{n_i}. As when setting oracle A, the limitation on the running times of M_j for $1 \le j \le i$ and the values of n_j for $1 \le j \le i$ ensure that the probability that M_i^B accepts is only a function of questions of the form $0x$ where x is of length n_i or $1x$ where x is of length $3n_i$. Define $p(b^{n_i}) = \Pr_r[M_i^B \text{ accepts } 1^{n_i}]$ and as before we have two cases.

1. There exists a redundant b^{n_i} such that $p(b^{n_i}) > 2^{-n_i}$. In this case we set B according to this b^{n_i}. That is, 1^{n_i} is not in L'_B but M_i^B fails to strong BPP reject it.
2. If no redundant b^{n_i} exists with $p(b^{n_i}) > 2^{-n_i}$ find a unique b^{n_i} with $p(b^{n_i}) \le \frac{1}{2}$ and set B accordingly. That is, 1^{n_i} is in L'_B but M_i^B fails to strong BPP accept it.

To verify the construction we need the following equivalent of Lemma 4.

Lemma 7: *If* $p(b^{n_i}) \le 2^{-n_i}$ *for all* $b^{n_i} \in R$ *then the fraction of* U *with* $p(b^{n_i}) < 1/2$ *is* $\ge 1 - 6 \cdot 2^{-n_i}$.

The proof of Lemma 7 is, of course, very similar to the proof of Lemma 4. However, there are some additional complications due to the dictionary. To take care of these problems we need the following definition.

Definition A machine M makes a *discovery* if on input 1^n it asks B about a string $1x, |x| = 3n, 1x \in B$ and it has not asked any oracle question(s) about a segment of b^n with value x.

We get the following lemma.

Lemma 8 *The probability that a probabilistic polynomial time machine M, which runs in time n^i makes a discovery is $\leq n^{i-1}/3 \cdot 2^{2n}$. The probability is taken over the coin flips of the machine and over a random b^n from either R or U.*

Proof: Fix r, consider any question "$1x \in B$" with $|x| = 3n$. The probability that x will be a segment of b^n which the machine has not yet seen is $\leq 1/3n2^{2n}$. Since M asks at most n^i questions the lemma follows.

$\square$

Let us use this lemma to prove Lemma 7. Consider the set of b^{n_i}, r which makes M_i^B accept. This set is divided into equivalence classes with pairs giving the same computation of M_i^B in the same equivalence class. We know that when b^{n_i} is given probabilities according to R the total mass of accepting conversations is at most 2^{-n_i}. When b^{n_i} is given proabilities according to U we have accepting conversations of two types: Computations where M_i^B makes a discovery and computations where it does not. By Lemma 8 the mass corresponding to the first case is bounded by $n_i^{i-1}/3 \cdot 2^{2n_i} \leq 2^{-n_i}$. Modifying the analysis of Lemma 5 slightly we see that the mass of the second type of conversations contributes at most twice the mass of the corresponding conversations with $b^{n_i} \in R$. Thus

$$\Pr_{r,b^n \in U}[M_i^B \text{ accepts } 1^{n_i}] \leq 2^{-n_i} + 2 \cdot 2^{-n_i} \leq 3 \cdot 2^{-n_i}$$

which implies that

$$\Pr_{b^n \in U}\left[\Pr_r[M_i^B \text{ accepts } 1^{n_i}] \geq \frac{1}{2}\right] \leq 6 \cdot 2^{-n_i}.$$

$\square$

4.2 Perfect Zero-Knowledge

To take care of the perfect zero-knowledge requirement we have to change the simulator.

Definition of simulator : On input 1^n, the simulator runs V' and produces a question c. Next the simulator checks if $1c \in B$. If not it returns "you cheat". If $1c \in B$ the simulator asks the oracle questions to obtain all segments of b^n in which V' has asked at least one question. If one of these segments is equal to c, $M_{V'}$ returns its index as the provers answer. If not $M_{V'}$ asks the 2^n oracle questions $0x \in B$ to locate the segment of b^n with value c and returns the position of the segment.

We need to check that this simulator runs in expected polynomial time for any verifier. Assume that the running time of V_j is n^j. The crucial lemma is:

Lemma 9: *The probability that M_{V_j} runs in time $\geq 3n^{j+2}$ is $\leq n^{j-1}/3 \cdot 2^{2n}$. Furthermore, M_{V_j} never runs for longer than $n2^n + 3n^{j+2}$ steps. The proability here is taken over random $b^n \in U$ and r, the random coins of the machine.*

Proof: We need only analyze how long M_{V_j} runs in the different cases. If $1c \notin B$ then M_{V_j} runs in time $n^j + n$. If c occurs as one of the segments that V_j looked at, the running time is at most $n^{j+2} + n^j + n \leq 3n^{j+2}$. In the last case we get the bound $n2^n + n^{j+2} + n^j + n$. To finish the proof we need only observe that to get into the last case the verifier has to make a discovery and hence we can use Lemma 8.

□

However we are more interested in probabilities over r for a fixed choice of b^n and this is taken care of by our last lemma.

Lemma 10: *For a fraction* $1 - 2^{-n}$ *of* U *the expected running time of* M_{V_j} *is bounded by* $4n^{j+2}$ *for all* j.

Proof: By Lemma 9 the fraction of U for which M_{V_j} has probability $\geq 2j^2n^{j-1}/3 \cdot 2^n$ of running in time $\geq 3n^{j+2}$ is $\leq 2^{-n}(2j^2)^{-1}$. Thus the fraction of U where this condition is violated for any j is bounded by $2^{-n}\sum_{i=j}^{\infty} \frac{1}{2j^2} \leq 2^{-n}$. If the condition is not violated, the expected running time is bounded by

$$3n^{j+2} + \frac{2j^2n^{j-1}}{3 \cdot 2^n} n2^n \leq 4n^{j+2}.$$

□

4.3 Interleaving the two conditions

Now combining section 4.1 and 4.2 to prove Theorem 2 is easy. We change condition $2'$ in the BPP construction to:

$2''$. If no redundant b^{n_i} with $p(b^{n_i}) > 2^{-n_i}$ exists find a unique b^{n_i} with $p(b^{n_i}) \leq \frac{1}{2}$ and such that M_{V_j} runs in expected time $4n_i^{j+2}$ for all j. Set B according to this b^{n_i}.

We just need to observe that, by Lemmas 7 and 10, a random b^{n_i} satisfies the two conditions with probability $\geq 1 - 7 \cdot 2^{-n_i} > 0$. A simple verification similar to that in section 3.3 finishes the proof of Theorem 2.

□

5. Remarks

We would like to remark here about the relative strength of PZK and SZK. Recall our relativized PZK language L'. Occassionally, the simulator was required to make an exponential search of $0x$ ($|x| = 2^n$) because the verifier produced a segment of b^n without making queries in the segment. Importantly, the simulator knew when it was necessary to make such a search (by looking in the dictionary) and the search was rarely necessary. This is precisely the same way in which all know SZK languages also have perfect zero-knowledge simulators (as noted in [GMR2]) The perfect simulator can determine when it is necessary to make an expontial time computation and the computation is rarely necessary.

However, our SZK language does not seem to have this property. When the verifier has produced a string but has not made queries in a segment of the oracle which looks like the string, the simulator has no way of knowing whether the string is a segment of b^n, i.e., whether to make an exponential search. It seems that in order for a simulation to be perfect it must make the exponential search whenever the above occurs but such a simulation certainly will not run in expected polynomial time.

Acknowledgments: We would like to thank Oded Goldreich for allowing us to use his idea of how to construct a relativized perfect zero-knowledge protocol.

References

[AGH] Aiello, W., S. Goldwasser, and J. Hastad, "On the Power of Interaction," to appear in *Combinatorica.* A preliminary version appeared in *Proc. of 27th Symposium on Foundations of Computer Science*, pp 368–379, Toronto, 1986.

[AH] Aiello, W. and J. Hastad. "Statistical Zero-Knowledge Languages Can Be Recognized in Two Rounds", to appear in *JCSS.* A preliminary version appeared in *Proc. of 28th Symposium on Foundations of Computer Science*, pp 439–448, Los Angeles, 1987.

[Ba] Babai, L., "Trading Group Theory for Randomness," *Proc. of 17th Symposium on Theory of Computing*, pp 421–429, Providence, 1985.

[BGGHKMR] Ben Or, M., O. Goldreich , S. Goldwasser, J. Hastad, J. Kilian, S. Micali, and P. Rogaway, "Everything Provable is Provable in Zero-Knowledge", *Proc. of Crypto '88,* Santa Barbara, 1988.

[BHZ] Boppana, R. B., J. Hastad, and S. Zachos, "Does *co-NP* Have Short Interactive Proofs?" *Information Processing Letters*, **25** (1987), pp 127–132.

[F] Fortnow L., "The Complexity of Perfect Zero-Knowledge," *Proc. of 19th Symposium on Theory of Computing*, pp 204–209, New York, 1987.

[GMR1] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proofs," *Proc. of 17th Symposium on Theory of Computing*, pp 291–305, Providence, 1985.

[GMR2] Goldwasser, S., S. Micali, and C. Rackoff, "Proofs, Knowledge, and Computation," **18** (1989), No. 1, pp 186–208.

[GMW] Goldreich, O., S. Micali, and A. Wigderson, "Prooof that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design," *Proc. of 27th Symposium on Foundations of Computer Science*, pp 174–187, Toronto, 1986.

[GS] Goldwasser, S., and M. Sipser, "Private Coins Versus Public Coins in Interactive Proof Systems," *Proc. of 18th Symposium on Theory of Computing*, pp 59–68, Berkeley, 1986.

[IY] Impagliazzo, R., and Moti Yung, "Direct Minimum-Knowledge Computations," *Proc. of CRYPTO '87*, pp 40–51, Santa Barbara, 1987.

[O] Oren, Y. "On the Cunning Powers of Cheating Verifiers: Some Observations about Zero-Knowledge Proofs", *Proc. of 28th Symposium on Foundations of Computer Science*, pp 462–471, Los Angeles, 1987.

[S] Shamir, Adi, "$IP = PSPACE$," manuscript.

[TW] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information," *Proc. of 28th Symposium on Foundations of*

Computer Science, pp. 472–482, Los Angeles, 1987.