Perfect Zero Knowledge: New Upperbounds and Relativized Separationss

Peter Dixon, Sutanu Gayen, A. Pavan, and N. V. Vinodchandran

***Abstract*—We investigate the complexity of problems that admit perfect zero-knowledge interactive protocols and establish new unconditional upper bounds and oracle separation results. We establish our results by investigating certain distribution testing problems: computational problems over high-dimensional distributions represented by succinct Boolean circuits. A relatively less-investigated com- plexity class SBP emerged as significant in this study. The main results we establish are:**

A unconditional inclusion that NIPZK $\subseteq$ CoSBP.

Construction of a relativized world in which there is a distribution testing problem that lies in NIPZK but not in SBP, thus giving a relativized separation of NIPZK (and hence PZK) from SBP.

Construction of a relativized world in which there is a distribution testing problem that lies in PZK but not in CoSBP, thus giving a relativized separation of PZK from CoSBP.

Results (1) and (3) imply an oracle separating PZK from NIPZK. Our results refine the landscape of perfect zero-knowledge classes in relation to traditional complexity classes.

A. *We obtain Theorem 1 by showing that Uniform is in CoSBP.*

Note that we can obtain relativized versions of the distribution testing problems by providing oracle access to the circuits involved. To obtain Theorem 2, we consider a promise problem that is a variant of Uniform.

Uniform-Or-Small: Given a distribution D, ΠY es = {⟨D⟩ — D = U } and ΠNo = {⟨D⟩ — —sup(D)— ≤ 2n/2} We show that a relativized version of this problem is not in SBP. For Theorem 3, we consider a variant of SD called Disjoint-Or-Identical.

Disjoint-Or-Identical: Given two samplable distributions C and D, ΠY es = {⟨C, D⟩ — sup(C)∩sup(D) =

∅} and ΠNo = {⟨C, D⟩ — C = D} (i.e, the distance between C and D is either 1 or 0).

This problem can be shown to be in CoPZK. We construct an oracle relative to which this problem is not in SBP. Theorems 2 and 3 show that there exist relativized worlds where PZK is neither in SBP nor in CoSBP. This suggests that we cannot hope to improve the containment PZK ⊆ PP to either SBP or CoSBP using relativizable techniques.

Notation and Definitions

Distributions. All the distributions considered in this paper are over a sample space of the form {0, 1}n for some integer n. Given a distribution D, we use D(x) to denote the probability of x with respect to D. We use Un to denote the uniform distribution over {0, 1}n. We consider distributions sampled by circuits. Given a circuit C mapping m-bit strings to n-bit strings, the distribution encoded/sampled by the circuit C is the distribution C(Um). We often use C to denote both the circuit and the distribution sampled by it. Note that given access to the circuit, we can efficiently generate a sample of the distribution by evaluating C on a uniformly chosen m-bit string. For this reason, we call such distributions efficiently samplable distributions or just samplable distributions. We use sup(D) to denote the set of strings for which D(x) /= 0.

Given two distributions C and D over the same sample space S, the statistical distance between them, denoted by dist (C, D), is defined as follows.

dist (C, D) = max(C(T) − D(T)) =

T ⊆S

(C(x) − D(x))

C(x)¿D(x)

Complexity Classes We refer the reader to the textbook by Arora and Barak [4] for definitions of stan- dard complexity classes. For a complexity class C, CoC denotes the class of complement languages/promise problems from C. The class SBP was introduced in [8] and is defined as follows.

Definition 1. A promise problem (ΠY es, ΠNo) is said to belong to the complexity class SBP if there exists a constant ϵ ¿ 0, a polynomial p(·), and a probabilistic polynomial-time Turing Machine M such that

If x ∈ ΠY es then Pr[M accepts] ≥ 1+∈

If x ∈ ΠNo then Pr[M accepts] ≤ 1 ,

SBP is sandwiched between MA and AM and is the largest known subclass of AM that is in PP. In fact, it is known that SBP is contained in the class BPPpath which is a subclass of PP.

Theorem 4 ([8]). MA ⊆ SBP ⊆ AM and SBP ⊆ BPPpath ⊆ PP.

Although we will not be using explicit definitions of zero-knowledge classes, we give necessary definitions for completeness.

Definition 2 (Non-Interactive protocol). A non-interactive protocol is a pair of functions ⟨P, V ⟩, the prover and verifier. On input x and random strings rI, rP , P sends a message π = P (x, rP , rI) to V , and

V computes m = V (x, π, rI). V accepts x if m = 1, and rejects if m = 0. The transcript of the interaction is the tuple ⟨x, rI, π, m⟩.

Note that the above definition implies that the random string rI is shared between the prover and the verifier.

Definition 3 (NIPZK[21, 16]). A promise problem ⟨ΠY es, ΠNo⟩ is in NIPZK (Non-Interactive Perfect Zero Knowledge) if there is a non-interactive protocol ⟨P, V ⟩ where V runs in polynomial time, and a ran- domized, polynomial-time computable simulator S, satisfying the following conditions:

(Soundness:) For any function P $*$ and any x $\in$ ΠNo, Pr[V accepts] $\leq$ 1/3

(Completeness:) If x $\in$ ΠY es, Pr[V accepts] $\geq$ 2/3

(Zero Knowledge:) For any x $\in$ ΠY es, the distribution of S(x) is identical to the distribution of the transcript generated by ⟨P, V ⟩ on input x.

The class NISZK (Non-Interactive Statistical Zero Knowledge) is defined similarly [16], except that we only require that the statistical distance between the distribution of S(x) and the distribution of the tran- script generated by ⟨P, V ⟩(x) be less than 1/p(n) for every polynomial p(n). Malka [21] showed that the promise problem Uniform is complete for the class NIPZK.

Theorem 5 ([21]). The promise problem Uniform is complete for NIPZK.

NIPZK $\subseteq$ CoSBP

For a given distribution D, let CP(D) denote the collision probability: Prx,y∼D(x = y). The following lemma is folklore. See [15] for a proof.

2

Lemma 1. For a given distribution D over {0, 1}n, if dist (D, Un) $\geq \epsilon$, then CP(D) $\geq$ 1+∈

Theorem 1. NIPZK $\subseteq$ CoSBP

We show the result by proving that the NIPZK-complete problem Uniform is in CoSBP. We start with the following lemma.

Lemma 2. Let D be a distribution on n + 1 bits, and let T = {x $\in$ {0, 1}n — x1 $\in$sup(D)}. Suppose that

—T — $\leq$ 2n/3 and Pr(D[n + 1] = 1) = 1 + ϵ for some ϵ $\geq$ 0. Then dist (D[1 . . . n], Un) is at least ϵ. Proof. Recall that dist (D[1 . . . n], Un) = maxS⊆{0,1}n Prd∼D[1...n][d $\in$ S] $-$ Pru∼Un [u $\in$ S]

max n Pr [d $\in$ S] $-$ Pr [u $\in$ S] $\geq$ Pr [d $\in$ T] $-$ Pr [u $\in$ T]

S⊆{0,1}

d∼D[1...n]

u∼Un

d∼D[1...n]

$\geq$ 1 + ϵ $-$ Pr

u∼Un

[u $\in$ T]

3 u∼Un

3 2n

1 2n n

$\geq$ + ϵ $-$ /2 = ϵ

3 3

Now we prove Theorem 1 by giving a CoSBP algorithm for Uniform.

Proof. Recall the definition of Uniform: Given a circuit D : {0, 1}m → {0, 1}n+1, ΠY es = {D : D[1 . . . n] =

Un, Pr[D[n + 1] = 1] $\geq$ 2/3} and ΠNo = {D : —sup(D) ∩ {0, 1}n1— $\leq$ 2n/3}.

Consider the following randomized algorithm: Given D as input, get two samples d0 and d1 from D. If the first n bits of both d0 and d1 are the same, then accept. Else, obtain k additional samples from D, and if the last bit of all these samples is 0, then accept, otherwise reject.

If D is a 'yes' instance of Uniform, then the probability of accepting at the first step is 1 and the probability of accepting at the second step is at most 1 , so the overall accept probability is $\leq$ 1 + 1 .

3k 2n 3k

Suppose that D is a 'no' instance of Uniform. By lemma 2, either D[1 . . . n] is at least 1 away from Un, or

1 6

D[n + 1] is 1 with probability at most 2 . Suppose that D is at least 1/6 away from the uniform distribution,

then by Lemma 1, the probability that the first n bits of d0 and d1 are the same is at least 37 1 . Thus

the algorithm accepts with probability at least 37 1 . Now suppose that D is less than 1/6 away from the uniform distribution. This implies that the last bit of D is 1 with probability at most 1/2. Thus in this case the algorithm accepts with probability $\geq$ 1 . Thus, a no instance is accepted with probability $\geq$ min 37 1 , 1 .

Choose k = n $-$ log(37/36), so that a no instance is accepted with probability $\geq$ 37 1 and a yes instance is

accepted with probability $\leq$ 1 + 3log(37/36) . For large enough n, 37 1 $\geq$ (1 + 1)(1 + 3log(37/36)), so this is

2n 3n

a CoSBP algorithm for for Uniform.

36 2n

40 2n 3n

Oracle Separations

In this section, we prove Theorems 2 and 3. We first prove a general approach that can be used to construct relativized worlds where promise problems involving circuits are not in SBP.

Lemma 3. Let Π = ⟨ΠY , ΠN ⟩ be a promise problem whose instances are circuits. If there is an oracle circuit family {Cn}n$\geq$0 and a constant c ¿ 1 with the following properties:

Cn is a oracle circuit that maps n bits to n bits and makes oracle queries only to strings of length cn.

There exist families of sets {An}n$\geq$0, {Bn}n$\geq$0 $\subseteq$ {0, 1}cn such that for all n, CAn $\in$ ΠY and CBn $\in$ ΠN

n n

For every probabilistic polynomial-time Turing Machine M and infinitely many n, for every Di $\in$

{Ai, Bi, ∅}, 1 $\leq$ i ¡ n

(∪n$-$1D)∪A

(∪n$-$1Di)∪An

Pr[M

i=1 i

n (Cn i=1

n$-$1

)accepts]

¡ 2,

Pr[M (∪n$-$1Di)∪Bn (C(∪i=1 Di)∪Bn)accepts]

i=1 n

then there exists an oracle O such that ΠO /$\in$ SBPO Proof. We first note that in this definition of SBP, we can choose ϵ to be 1 by using amplification techniques. Thus a promise problem is in SBP if there exists a polynomial p(·) and a probabilistic polynomial-time machine M such that on positive instances M accepts with probability at least 2/2p(n) and on negative instances M accepts with probability at most 1/2p(n). We call p(·) the threshold polynomial for M .

Let {Mi}i¿0 be an enumeration of the probabilistic polynomial-time machines. We consider an enumera- tion of tuples ⟨Mi, j⟩i¿0,j¿0. In this enumeration considering ⟨Mi, j⟩ corresponds to the possibility that Mi is a SBP machine with threshold polynomial nj. We first start with an empty oracle. Let Oi = O ∩ {0, 1}ci. For each i, Oi will be one of ∅, Ai or Bi. Consider ⟨Mi, j⟩ and let n be a length for which On is not yet defined and for which the inequality from the lemma holds for the machine Mi. Suppose that Mi makes queries of length ≤ m. Note that by this, we have defined Oi for all i ¡ cn, thus O ⊆ {0, 1}¡cn and for every i ¡ n Oi

is either ∅, A or B . Suppose that the acceptance probability of MO∪An (CAn nj

i i i

) is less than 2/2

. We set O

at length cn as An and for all the lengths from cn + 1 to m the oracle O is set to be ∅. Now CAn is a positive instance for which Mi cannot be a SBP machine with nj as the threshold polynomial. Then we set O at length cn as An and move to the next tuple in the enumeration. Suppose that MO∪An (CAn) accepts with

probability at least 2/2nj

nj

. Now by the inequality from lemma 3, the acceptance probability of MO∪Bn (CBn)

Bn

is more than 1/2

. Note that C

is a negative instance for which Mi is not a SBP machine with threshold polynomial nj. Thus we make the oracle O at length cn to be Bn. It is easy to see that ΠO is not in SBPO: Suppose not, and there exists a probabilistic polynomial-time machine Mi with threshold polynomial nj. When we considered the tuple ⟨Mi, j⟩, we ensured that Mi does not have threshold polynomial nj on COcn .

Oracle Separation of NIPZK from SBP

In this section we show that Theorem 2 cannot be improved to show that NIPZK is a subset of SBP using relativizable techniques. For this we show that the oracle version of Uniform-Or-Small is not in SBP.

Theorem 6. There exists an oracle O relative to which Uniform-Or-Small is not in SBPO.

Malka [21] showed that Uniform-Or-Small is in NIPZK, and this proof relativizes. Combining this with Theorem 6, we obtain Theorem 2. To prove Theorem 6, it suffices to exhibit sets An and Bn that satisfy the conditions of Lemma 3. We construct these sets via a probabilistic argument. We first provide a brief overview of this construction.

Remark: There is a alternate proof of the oracle separation between NIPZK from SBP which we describe here briefly. This was pointed out to us by one of the reviewers of TCC 2020. The proof uses known facts about the well-studied Permutation Testing Problem (PTP). PTP takes as input a truth table of a function f : [N] → [N] promised to be either a permutation on [N] or N/3 away in Hamming distance from any permutation on [N]. The computational goal is to distinguish these two cases. It is known that in the query-complexity setting, there is a NIPZK protocol where the verifier uses public randomness to pick a uniform random element x from [N], which is viewed as an element from the range of the function, and the prover is required to present a preimage of x. Aaronson, in [1] (Theorem 13), gave the construction of an oracle separating SZK from the Quantum version of SBP using degree arguments. The oracle is derived from the PTP problem where the author uses a SZK upper bound for PTP. However, as noted above the upper bound of NIPZK holds for PTP and hence it gives an oracle separation of NIPZK from SBP. Here we provide an oracle separation using elementary arguments.

Overview of the proof: Consider a non-relativized world with the following restriction on how a probabilistic polynomial-time machine M can access the input circuit C: At the beginning the machine gets to see a sequence S of k independent samples from C. After this the machine ignores C. Note that in this model the underlying machine cannot perform adaptive sampling from C, nor can the machine generate samples that might be correlated. In this model it is easy to see that if C encodes the uniform distribution, the probability that M is presented with a specific sequence S of k samples is precisely 1/2nk. Thus the probability that the machine M accepts is S Pr[M accepts S] , summed over all sequences of size k.

Now given a subset D of {0, 1}n of size 2n/2, let UD be the uniform distribution over D. Consider the following experiment. Randomly pick D and let CD be a circuit that samples UD. Independently draw a sequence of k samples S from UD and present them as input to M . (In a non-relativized setting, there may not be a small circuit that uniformly samples D, but in the relativized worlds we consider, this is not an issue.) We consider the acceptance probability of M over random choices of D, S and internal coin tosses of

M . By a careful analysis we can show that this probability is very close to S Pr[M accepts S] . Thus the ratio between the acceptance probabilities of M when given samples from the uniform distributions and samples drawn from UD (over a random choice of D) is less than $1 + \epsilon$ for any constant ϵ. By a probabilistic

argument, there exists a subset D such that the acceptance probability of M on a positive instance (U) and a negative instance (UD) are the same. Thus M is not a SBP machine.

The crux of the above idea is that when the samples are generated independently and nonadaptively, then it is possible to argue that a SBP machine cannot distinguish between whether they came from the uniform distribution or from a distribution with small support size. Now, we need to argue in the more general model, where a probabilistic machine can do adaptive sampling and generate samples that could be correlated to each other. A first approach to construct the sets An and Bn is to encode the uniform distribution in An and the distribution UD in Bn. The set An can be defined as {⟨i, j⟩ — the ith bit of the jth string of Σn is 1} (in the standard lexicographical ordering). To define Bn given D, first consider the multiset D that contains 2n/2 copies of each elements of D. Thus the cardinality of D is 2n. Now, the set Bn can be defined as tuples

⟨l, j⟩ where the lth bit of the jth string of D is 1. Consider the oracle circuit C which is defined as follows:

Definition 4 (Oracle Circuit). Let CO be a fixed linear-size oracle circuit, with n inputs and n outputs, defined as follows: On input j ∈ {1 . . . 2n}, CO(j) outputs O(⟨l, j⟩) for all l between 1 and n. In other words, CO(j) outputs the jth string of O.

Notice that CAn is the uniform distribution and CBn is uniform on D and the goal of the probabilistic machine is to distinguish between the distributions CAn and CBn . However, if we allow correlated sampling, a probabilistic machine can easily distinguish CAn and CBn by computing CO(j) and CO(j +1) for appropriate inputs j and j + 1 and comparing whether they are equal or not. To guard against such behavior, we apply one more level of randomization - randomize the underlying order of the strings. Thus the tuple ⟨l, j⟩ will encode the lth string in an order that is not necessarily the standard lexicographic order. We argue that when we randomly order {0, 1}n, then adaptive and correlated sampling does not give significantly more information than independently generated samples. Now, we proceed to give a formal proof.

Detailed proof: From now on, we fix a length n. We use a probabilistic argument to construct An and Bn. For An we consider 2n! sets Yi and define An to be one of them (using a probabilistic argument), and similarly for Bn we consider many sets NDi and define Bn to be one of them.

Definition 5 (Oracle families). Let 1 ≤ i ≤ 2n! index the set of all 2n! permutations of {0, 1}n. Oracles for Yes instances: Yi = {⟨l, j⟩: the lth bit of the jth string of the ith permutation of {0, 1}n is 1}.

Oracles for No instances: For each set D of size d = 2m (where m = n/2) let D be the multiset that contains

2n−m copies of each element of D. Thus —D— = 2n, and we define ND as: ND = {⟨l, j⟩: the lth bit of the

i

jth string of the ith permutation of D is 1}.

For the rest of this section, we will use Y to represent an arbitrary Yi oracle, N to represent an arbitrary ND oracle, and O to represent an arbitrary Yi or ND . Note that for every i, CYi is the uniform distribution

i i

and CNDi is the uniform distribution on D and thus has small support.

We first prove the following lemma and show later how to build on it to arrive at the conditions specified in Lemma 3.

Lemma 4. If i is uniformly chosen from {1, . . . , 2n!} and D is uniformly chosen from all size 2m subsets of

{0, 1}n, then for any constant c ¿ 1 and every probabilistic polynomial-time algorithm A, for large enough

n,

Pri,r[AYi accepts CYi]

≤ c

Pr [ANDi accepts CNDi]

where r is the random choice of A.

Without loss of generality we can assume that any oracle query that AO makes can be replaced by evaluating the circuit CO, by modifying A in the following way: whenever A queries the oracle O for the ith bit of the jth string, it evaluates CO(j) and it extracts the ith bit. We refer to this as a circuit query. Let k be the number of circuit queries made by A, where k is bounded by a polynomial. We will use q1, . . . qk to denote the circuit queries, and denote the output CO(qi) by ui. We can assume without loss of generality that all qi are distinct. We use S to denote a typical tuple of answers ⟨u1, · · · , uk⟩. We will use AS to denote the computation of algorithm A when the answers to the circuit queries are exactly S in that order. Notice that the AS does not involve any oracle queries. Once A has received S, it can complete the computation without any circuit queries. So, the output of AS is a random variable that depends only on the internal randomness r of A.

Claim. Without loss of generality we can assume that along any random path, A rejects whenever any

ui = uj, i /= j.

Proof. In a Yes instance, CY is uniform. Since C has n inputs and n outputs, CY is a 1-1 function. By the earlier assumption, ui will never match any other uj. In a No instance, CN will have 2n−m inputs for any output. Rejecting any time ui = uj will not affect Pr[A accepts a Yes instance], and it will re- duce Pr[A accepts a No instance]. Thus the ratio of the probability of accepting an Yes instance and the probability of accepting a No instance only increases. We will show that this higher ratio is ¡ c.

We will use the following notation.

"AO asks ⟨q, i⟩" is the event that "the ith circuit query made by A is CO(q)." For simplicity, we write this event as "AO asks qi."

"AO gets ⟨u, i⟩" is the event that "CO(q) = u where q is the ith query". Again, for simplicity, we write this event as "AO gets ui."

For S = ⟨u1, . . . uk⟩, "AO gets S" is the event that "AO gets u1 and AO gets u2 and . . . AO gets uk (in that order)".

Lemma 5. For any probabilistic algorithm A and for any fixed S = ⟨u1, . . . uk⟩ where all ui are distinct,

Pr[AYi gets S and accepts] = Pr[A accepts] Y 1

Proof.

Pr[AYi gets S and accepts] = Pr[AYi gets S] × Pr[AYi accepts —AYi gets S]

i,r

i,r

i,r

= Pr[AYi gets S] × Pr[AS accepts]

i,r r

The last equality is because AS is independent of i as discussed before. We will show that Pri,r[AYi gets S] =

Qk−1 1 which will prove the lemma.

k−1

Pr[AYi gets S] = Pr[AYi gets uj+1—AYi gets ⟨u1, u2, . . . , uj⟩]

For any fixed j let Ej denote the event "AYi gets ⟨u1, u2, . . . , uj⟩". Then,

Pr[AYi gets uj+1—AYi gets ⟨u1, u2, . . . , uj⟩] = Pr[AYi gets uj+1—Ej]

i,r

i,r

= Σ Pr[AYi asks qj+1—Ej] × Pr[CYi (qj+1) = uj+1—Ej]

qj+1

= Σ Pr[AYi asks qj+1—Ej] × Pr[CYi (qj+1) = uj+1—Ej]

= Σ Pr[AYi asks q —E] × 1

= 1 × Σ Pr[AYi asks q —E]

(2n − j)

1

= (2n − j)

qj+1

i,r

j+1 j The third equality is because the output of C is independent of r and the fourth equality follows from the fact that for a random permutation of {0, 1}n, once j elements are fixed, there are 2n − j equally likely possibilities for uj+1. The lemma follows.

Lemma 6. For any algorithm A and any fixed S = ⟨u1, . . . uk⟩ where uis are distinct,

k−1 m

n−m

Pr [ANDi gets S and accepts] = Pr[A

accepts] × Y (2 − j)2

Proof. The argument is identical to the proof of Lemma 5 except for the probability calculations.

Pr [ANDi gets S and accepts] = Pr [ANDi gets S] × Pr [ANDi accepts —ANDi gets S]

i,r,D

i,r,D

i,r,D

= Pr [ANDi gets S] × Pr[AS accepts]

i,r,D r

The last equality is because AS is independent of i and D. We will show that Pri,r,D[ANDi gets S] =

Qk−1 (2m−j)2n−m which will prove the lemma.

k−1

Pr [ANDi gets S] = Pr[ANDi gets uj+1—ANDi gets ⟨u1, u2, . . . , uj⟩]

We will reuse the notation Ej for convenience. For any fixed j, let Ej denote the event "ANDi gets ⟨u1, u2, . . . , uj⟩" Then,

Pr [ANDi gets uj+1—ANDi gets ⟨u1, u2, . . . , uj⟩] = Pr [ANDi gets uj+1—Ej]

i,r,D

i,r,D = Σ

Pr [ANDi asks qj+1—Ej] × Pr [CNDi (qj+1) = uj+1—Ej]

qj+1

= Σ Pr [ANDi asks qj+1—Ej] × Pr[CNDi (qj+1) = uj+1—Ej]

We will show that for any q, Pr

[CYi (q) = u

—E] = (2m−j)2n−m .

i,D

j+1 j

(2n−j)2

Pr[CNDi (q) = uj+1—Ej] = Pr[uj+1 ∈ D—Ej] × Pr[CNDi (q) = uj+1—uj+1 ∈ D, Ej]

i,D

i,D

2n−j−1

n−m

i,D

= 2n−j 2m−j

2m − j

2n − j

2n−m

= 2n − j × 2n − j

(2m − j)2n−m

= (2n − j)2

The second equality is because of the following reasoning. There are 2n−j choices of D where u . . . uj

are included, and 2n−j−1 that include uj+1 as well. Given that u1, . . . uj+1 ∈ D, the probability that

n−m

CNDi (qj+1) = uj+1 is 2 (since there are 2n−m copies of uj+1 remaining, and 2n−j total things remaining).

We need the following claim.

Claim. For any polynomial k = k(n) and any constant c ¿ 1, for large enough n,

k−1 n

2 − j ¡ c

2n − 2n/2j

j=0

Proof.

k−1 n k−1 n

Y 2 − j ≤ Y 2

j=0

2n − 2n/2j

j=0 k−1

2n − 2n/2j

n/2

= 2

2n/2 − j

j=0

2n/2 k

≤ 2n/2 − k

k k

= 1 + 2n/2 − k

For any polynomial k = k(n), limn→∞(1 + k(n))k(n) = 1. Hence the claim.

We can now prove Lemma 4.

Proof (Proof of Lemma 4).

From lemmas 5 and 6, we have

Pri,r[AYi accepts CYi]

ΣS Pri,r[AYi gets S and accepts]

= Σ

Pri,r,D[ANDi accepts CND] Pri,r,D[ANDi gets S and accepts]

Σ Pri,r[AYi gets S and accepts]

≤ S distinct

S dΣistinct Pri,r,D[ANDi gets S and accepts]

Σ Pr [A accepts] × Qk−1 1

Σ Pr [A accepts] × Qk−1 (2m−j)2n−m

k−1 n

= 2 − j (substituting m = n/2) 2n − 2n/2j

j=0

¡ c (by Claim 4.1)

The second equality follows because when the oracle is Yi, S is always disjoint (as we never ask the same query twice) and when the oracle is NDi we assume that the algorithm rejects when S is not distinct.

(Completing the proof of Theorem 6): We will construct an oracle so that conditions of Lemma 3 are met. By a probabilistic argument, there exists an i∗ and D∗ such that

Pr[AYi∗ accepts CYi∗]

ND∗

ND∗

] ¡ c

Pr[A i∗ accepts C i∗

for every c ¿ 1 (by Lemma 4). Now define An as Yi∗ and Bn as ND∗ . This looks very close to the conditions of Lemma 3 except that we restricted the oracles to be An and Bn, However, for Lemma 3, we require that oracles are of the form (∪n−1Di ∪ An) and (∪n−1Di ∪ Bn). To establish this, we resort to the standard techniques

i=1 i=1

used in oracle constructions. Observe that the sets An and Bn can be constructed in double exponential

nj−1

time. Let n1 = 2 and nj = 22 . We will satisfy the conditions of Lemma 3 at lengths of the form n . For

nj−1

∪nj−1 D ∪A

every i that is not of the form n , we set both A and B to empty. Now M ∪i=1 Di∪Anj (C i=1 i nj) can

j i i nj

be simulated using MAnj (CAnj). As for queries whose length does not equal c · nj, the machine can find answers to oracle queries without actually making the query.

Oracle Separation of PZK from CoSBP In this section we construct an oracle that separates PZK from CoSBP, thus proving Theorem 3. For this we exhibit an oracle where the promise problem Disjoint-Or-Identical is not in SBP. This problem is a generalization of graph non-isomorphism (GNI) problem, in the sense that GNI reduces to this problem. Let G1 and G2 be two graphs, and let Ci be the distribution obtained by randomly picking a permutation π and outputting π(Gi). Observe that if G1 and G2 are not isomorphic then the supports of C1 and C2 are disjoint, and if G1 is isomorphic to G2, then C1 = C2. Moreover the distributions C1 and C2 can be sampled by polynomial-size circuits. The PZK protocol for graph isomorphism can be adapted to show that Disjoint-Or-Identical is in CoPZK.

Theorem 7. Disjoint-Or-Identical is in CoPZK Theorem 3 follows from the following theorem.

Theorem 8. There exists an oracle O relative to which Disjoint-or-Identical is not in SBPO

Input presentation: In the definition of Disjoint-Or-Identical, the input instances are tuples consisting of two circuits. However, we will represent them as just one circuit C in the following manner. Given a circuit C, let C0 denote the circuit obtained by fixing the first input bit of C to be 0, and the circuit C1 denote the circuit obtained by fixing the first input bit of C to be 1. An input to Disjoint-Or-Identical will be a circuit C and the goal is to distinguish between the cases “the support of distributions C0 and C1 are disjoint” or “C0 and C1 are identical distributions”.

The proof structure of this result is similar to that of Theorem 6 and as in that case, the goal is to construct a circuit family Cn and families of sets An and Bn that satisfy the conditions of Lemma 3.

Definition 6 (Oracle families). Let i ∈ {1 . . . 2n } index the partitions of {0, 1}n into two sets S0 and

i i i

Oracles for Yes instances: Yijk is an oracle for the set {⟨0, l, m⟩ : the lth bit of the mth string in the jth permutation of S0 = 1} ∪ {⟨1, l, m⟩ : the lth bit of the mth string in the kth permutation of S1 = 1}.

i i

Oracles for No instances: We construct the No instances similarly, except both 0 and 1 cases query S0. That is, Nijk is an oracle for the set {⟨0, l, m⟩ : the lth bit of the mth string in the jth permutation of

S0 = 1} ∪ {⟨1, l, m⟩ : the lth bit of the mth string in the kth permutation of S0 = 1}

i i

An oracle of the above form will be denoted by O which is a disjoint union of sets denoted by O0 and

O1. Now we define the input circuits that sample the two distributions.

Definition 7 (Oracle circuits). Let CO be a fixed linear-size oracle circuit, with n+1 inputs and n outputs, defined as follows: on input ⟨0, j⟩ where j ∈ {1 . . . 2n}, CO(j) outputs O0(⟨l, j⟩) for all l between 1 and n, and on input ⟨1, j⟩ where j ∈ {1 . . . 2n}, CO(j) outputs O1(⟨l, j⟩) for all l between 1 and n. In other words, CO(⟨0, j⟩) outputs the jth string of O0 and CO(⟨1, j⟩) outputs the jth string of O1.

We will establish the following lemma. Then the proof of Theorem 8 follows by arguments identical to that of the previous oracle construction.

Lemma 7. If i, j, k are uniformly and independently chosen from {1 . . . 2n }, {1 . . . 2n−1!}, {1 . . . 2n−1!} respectively, then for any probabilistic polynomial-time algorithm A, for any constant c ¿ 1, for large enough n,

Pri,j,k,r[AYijk accepts CYijk]

≤ c
Pri,j,k,r[ANi,j,k accepts CNi,j,k]
We use the same notation and make the most of same simplifications from the previous construction, with the following differences. The first difference is: let h be the (polynomial) maximum number of queries made by an algorithm A for any random choice of i, j, k, r. We will allow A to make 2h queries, two at a time, with the restriction that one must begin with 0 and the other must begin with 1. Notationally, p1 . . . ph are the queries that begin with 0 and ui is the result of query pi. q1 . . . qh are the queries that begin with 1 and vi is the result of query qi. S is the ordered multiset ⟨u1, v1, . . . uh, vh⟩. Notice that this is without loss of generality as A can simulate the original algorithm by ignoring either qi or pi as appropriate. The second difference is that, instead of assuming A rejects if any ui matches any uj, we assume A rejects if any ui matches any vj.
Lemma 8. For any probabilistic algorithm A and for any fixed S = ⟨u1, v1, . . . uh, vh⟩ where all elements of
S are distinct,
Pr [AYijk gets S and accepts] = Pr[A
accepts] × Y 1
Proof. Note that
Pr
i,j,k,r
[AYijk gets S and accepts] = Pr
i,j,k,r
[AYijk gets S] × Pr
i,j,k,r
[AYijk accepts — AYijk gets S]
= Pr [AYijk gets S] × Pr[AYijk accepts]
i,j,k,r r S
Thus we need to prove that
Pr
i,j,k,r
h−1
[AYijk gets S] =
(2n − 2l)(2n − 2l − 1)
A=0
We use EA to denote the event AYijk gets ⟨u1, v1, · · · uA, vA⟩. Note that
and
Pr
i,j,k,r
h−1
[AYijk gets S] =
A=0
Pr
i,j,k,r
[AYijk gets uA+1, vA+1 — EA]
Pr
i,j,k,r
[AYijk gets uA+1, vA+1 — EA] =
p,q
Pr
i,j,k,r
[AYijk asks pA+1 and qA+1—EA]
× Pr
i,j,k,r
[CYijk (p) = uA+1 and CYijk (q) = vA+1 — AYijk asks pA+1 and qA+1, EA]
Pr[A gets uA+1, vA+1—A asks pA+1, qA+1—EA]
= Pr[uA+1 ∈ S0, vA+1 ∈ S1—EA]
i i i
× Pr[uA+1 is the pth element of S0—EA]
A+1 i
× Pr[vA+1 is the qth element of Sh+1—EA]
A+1 i
2n − 2l − 2 , 2n − 2l 1 1
= 2n−1 − 1 − 1
1
2n−1 − 1
2n−1 − 1 2n−1 − 1
= (2n − 2l)(2n − 2l − 1)
Thus
Pr i,j,k,r
[AYijk gets uA+1, vA+1 — EA] =
p,q
Pr
i,j,k,r
[AYijk asks pA+1 and qA+1—EA]
× Pr
i,j,k,r
[CYijk (p) = uA+1 and CYijk (q) = vA+1 — AYijk asks pA+1 and qA+1, EA]
= 1 (2n − 2l)(2n − 2l − 1)
p,q
Pr
i,j,k,r
[AYijk asks p
A+1
and q
A+1
—EA]
1
= (2n − 2l)(2n − 2l − 1)
Since Pri,j,k,r[AYijk gets S] = Qh−1 Pri,j,k,r[AYijk gets uA+1, vA+1 — EA], using this with the above derived
equality we obtain that
Pr
i,j,k,r
h−1
[AYijk gets S] =
(2n − 2l)(2n − 2l − 1)
A=0
This completes the proof of the lemma. Now we turn to the No instances.
Lemma 9. For any algorithm A, for any fixed S = {u1, v1, . . . uh, vh} that are all distinct,
h−1 n n
Pr [ANijk gets S and accepts] = Pr[A
accepts] × Y (2 − 2l)(2 − 2l − 1) 1

i,j,k,r

Proof. As before,

r S

A=0

(2n−1 − 2l)(2n−1 − 2l − 1) (2n−1 − l)2

Pr

i,j,k,r

[ANijk gets S and accepts] = Pr

i,j,k,r

[ANijk gets S] × Pr

i,j,k,r

[ANijk accepts — ANijk gets S]

= Pr [ANijk gets S] × Pr[ANijk accepts]

It suffices to show that

i,j,k,r r S

h−1 n n

Pr

i,j,k,r

[ANijk gets S] = (2 − 2l)(2 − 2l − 1) 1

(2n−1 − 2l)(2n−1 − 2l − 1) (2n−1 − l)2

A=0

If EA denotes the event "ANijk gets ⟨u1, v1, ⟨, uA, vA⟩", then

Now,

Pr

i,j,k,r

h−1

[ANijk gets S] = Pr [ANijk gets uA+1, vA+1 — EA]

ijkr

A=0

Pr [ANijk gets uA+1, vA+1 — EA] = Σ Pr [ANijk asks pA+1 and qA+1 — EA]

× Pr [CNijk (p) = uA+1 and CNijk (q) = vA+1 — EA, ANijk asks pA+1 and qA+1]

ijkr

Consider the event "CNijk (p) = uA+1 and CNijk (q) = vA+1", conditioned on EA and "ANijk asks pA+1 and

qA+1". For this event to happen, it must be the case that both uA+1 and vA+1 are in S0, and uA+1 is the pth

i A+1

element of S0, and vA+1 is the qth element of S0. The probability that both uA+1 and vA+1 are in S0 given

i A+1 i i

that EA and A asks pA+1 and qA+1 is

2n − 2l − 2

, 2n − 2l

2n−1 − 2l

(2n−1 − 2l)(2n−1 − 2l − 1)

(2n − 2l)(2n − 2l − 1)

The probability that uA+1 is the pst element given EA is 1/(2n−1 − l) and similarly, the probability that

vA+1 is the qA+1st element given EA is 1/(2n−1 − l). Thus

Pr [AN

ijkr

ijk gets uA+1, vA+1 — EA] =

=

(2n−1 − 2l)(2n−1 − 2l − 1) (2n − 2l)(2n − 2l − 1)

(2n−1 − 2l)(2n−1 − 2l − 1) (2n − 2l)(2n−1 − 2l − 1)

1 (2n−1 − l)2

1

(2n − l)2

Σp,q

Pr [AN

ijkr

ijk asks pA+1 and qA+1 — EA]

Thus

h−1

n−1

n−1

Pr [ANijk gets S] = Y (2 − 2l)(2 − 2l − 1) 1 ,

i,j,k,r

and the lemma follows.

We need the following claim

A=0

(2n − 2l)(2n − 2l − 1)

(2n−1 − l)2

Claim. For any polynomial h = h(n) and any constant c ¿ 1, for large enough n,

h−1

n−1 2

(2 − l) ¡ c

(2n−1 − 2l)(2n−1 − 2l − 1)

A=0

Proof.

h−1

n−1 2

h−1

n−1 2

Y (2 − l) ≤ Y (2 − l)

A=0 A=0

h−1

≤

A=0

l + 1 2

1 + 2n−1 − 2l − 1)

For any polynomial h, the above expression tends to 1 for large enough n.

The rest of the proof of Lemma 7 and that of Theorem 8 is identical to the proofs of Lemma 4 and Theorem 6.

Acknowledgements

We thank the reviewers for their comments and suggestions. In particular, we thank an anonymous reviewer for pointing an alternate proof of Theorem 2 and making us aware of Aaronson's work [1] and also for pointing out Corollary 1.

References

Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. Quantum Inf. Comput., 12(1- 2):21–28, 2012.

Scott Aaronson, Baris Aydinlioglu, Harry Buhrman, John M. Hitchcock, and Dieter van Melkebeek. A note on exponential circuit lower bounds from derandomizing arthur-merlin games. Electronic Colloquium on Computa- tional Complexity (ECCC), 17:174, 2010.

William Aiello and Johan Hastad. Statistical zero-knowledge languages can be recognized in two rounds. Journal of Computer and System Sciences, 42(3):327 – 345, 1991.

Sanjeev Arora and Boaz Barak. Computational Complexity - A Modern Approach. Cambridge University Press, 2009.

Baris Aydinlioglu, Dan Gutfreund, John M. Hitchcock, and Akinori Kawachi. Derandomizing arthur-merlin games and approximate counting implies exponential-size lower bounds. Computational Complexity, 20(2):329– 366, 2011.

Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, pages 103–112. ACM, 1988.

Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. SIAM

J. Comput., 20(6):1084–1118, 1991.

Elmar Bo¨hler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. J. Comput. Syst. Sci., 72(6):1043–1076, 2006.

Ravi B. Boppana, Johan H°astad, and Stathis Zachos. Does co-np have short interactive proofs? Inf. Process. Lett., 25(2):127–132, 1987.

Adam Bouland, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the power of statistical zero knowledge. In Chris Umans, editor, 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017, pages 708–719. IEEE Computer Society, 2017.

L. Fortnow. The complexity of perfect zero-knowledge. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87, page 204–209, New York, NY, USA, 1987. Association for Computing Machinery.

Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In Li Gong and Michael K. Reiter, editors, CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA, November 3-5, 1998, pages 67–72. ACM, 1998.

Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. J. Comput. Syst. Sci., 60(3):540–563, 2000.

Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. J. ACM, 38(3):691–729, 1991.

Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. Electronic Colloquium on Computational Complexity (ECCC), 7(20), 2000.

Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 467–484. Springer, 1999.

S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems.

SIAM J. Comput., 18(1):186–208, 1989.

Greg Kuperberg. How hard is it to approximate the jones polynomial? Theory Comput., 11:183–219, 2015.

Shachar Lovett and Jiapeng Zhang. On the impossibility of entropy reversal, and its application to zero-knowledge proofs. In Yael Kalai and Leonid Reyzin, editors, Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I, volume 10677 of Lecture Notes in Computer Science, pages 31–55. Springer, 2017.

Lior Malka. How to achieve perfect simulation and a complete problem for non-interactive perfect zero-knowledge.

J. Cryptology, 28(3):533–550, 2015.

Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. J. Comput. Syst. Sci., 60(1):47– 108, 2000.

Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In David A. Wagner, editor, Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Con- ference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings, volume 5157 of Lecture Notes in Computer Science, pages 536–553. Springer, 2008.

Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. J. ACM, 50(2):196–249, 2003.

Thomas Watson. The complexity of estimating min-entropy. Computational Complexity, 25(1):153–175, 2016.