

Abstract—V can be simulated by a randomized polynomial time simulator S so that the output distribution of the simulator is statistically close to the distribution of the interaction. The intuition is that the interaction itself can be simulated efficiently (in randomized polynomial-time) and hence the verifier is not gaining any additional knowledge other than what she can simulate by herself. The class of problems that admit statistical zero-knowledge interactive proofs is denoted by SZK. An important restriction is when the output distribution of the simulator is identical to the distribution of the interaction. Such a protocol is called a perfect zero-knowledge protocol, and the corresponding class of languages is denoted by PZK [17]. It is also possible to envision a non-interactive situation where the only communication in the protocol is from the prover to the verifier. Indeed, Blum, Feldman and Micali [6] and De Santis et al [7] investigated such non-interactive zero-knowledge proofs and introduced the class NISZK. The corresponding perfect zero-knowledge class NIPZK was first investigated by Malka [21].

I. OUR CONTRIBUTIONS

One of our main contributions is a new unconditional upper bound on the complexity class NIPZK.

Theorem 1. $\text{NIPZK} \subseteq \text{CoSBP}$.

The class SBP (Small Bounded-error Probability) was introduced by Böhler, Glaßer, and Meister [8] and is a bounded-error version of PP. Informally, a language is in PP if there is a probabilistic polynomial-time machine for which the ratio between the acceptance and the rejection probabilities is more than 1 for all the positive instances. We obtain the class SBP when we stipulate that this ratio is bounded away from 1, i.e., $(1 + \epsilon)$ for a fixed constant $\epsilon > 0$. This restriction greatly reduces the power of the class. In particular, it is known that SBP is a subset of AM and PP, and contains MA. This class has been studied in other contexts as well, such as in circuit complexity and quantum computation [5, 2, 19, 1]. Even though the relationship between SBP and zero knowledge classes has not been studied earlier, a curious connection exists between them. Watson showed that a certain promise problem regarding the min-entropy of samplable distributions is a complete problem for SBP [25]. Interestingly, the analogous problem where entropy instead of min-entropy is considered was shown to be complete for the class NISZK [16]. Our upper bound result improves the known containments $\text{NIPZK} \subseteq \text{AM} \cap \text{coAM}$ to $\text{NIPZK} \subseteq \text{AM} \cap \text{CoSBP}$ and $\text{NIPZK} \subseteq \text{PP}$ to $\text{NIPZK} \subseteq \text{CoSBP}$.

We consider the possibility of establishing other upper bounds for perfect zero-knowledge classes. Since NIPZK is not known to be closed under complement, is it possible to show that $\text{NIPZK} \subseteq \text{SBP} \cap \text{CoSBP}$? We also consider whether we can show that PZK itself lies in CoSBP. For these two questions, we prove the following relativized lower bound results.

Theorem 2. There is an oracle O such that NIPZKO (and thus PZKO) is not in SBPO.

This result along with Theorem 1 implies that NIPZK is not closed under complement in a relativized world, a result that was recently established by Bouland et al. [10].

Theorem 3. There is an oracle O such that PZKO is not in CoSBPO.

As Theorem 1 relativizes with respect to any oracle, Theorem 1 and Theorem 3 together implies an oracle that separates PZK from NIPZK.

Corollary 1. There is an oracle O such that $\text{NIPZKO} \not\subseteq \text{PZKO}$.

Figure 1 summarizes the known relationships among perfect zero knowledge classes and other complexity classes along with the results established in this work.

PP
AM BPPpath CoAM
NIPZK CoNIPZK

Fig. 1: $A \rightarrow B$ indicates that A is a subset of B, $A \dashv B$ indicates that there is a relativized world where A

is not a subset of B. Red and Blue arrows indicate new results.

Complexity of Distribution Testing Problems. We establish our results by investigating certain distribution testing problems: computational problems over high-dimensional distributions represented by succinct Boolean circuits. Interestingly, it turns out that versions of distribution testing problems characterize various zero-knowledge classes. The distribution testing problems are best formalized as promise problems. A promise problem is a pair of sets $\Pi = (\Pi_{\text{Y es}}, \Pi_{\text{No}})$ such that $\Pi_{\text{Y es}} \cap \Pi_{\text{No}} = \emptyset$. $\Pi_{\text{Y es}}$ is called the set of ‘yes’ instances, and Π_{No} is called the set of ‘no’ instances. Given a Boolean circuit C mapping from m bits to n bits, the distribution sampled by C is obtained by uniformly choosing $x \in \{0, 1\}^m$ and evaluating C on x. We often use C itself to denote the distribution sampled by the circuit C.

Statistical Difference (SD): Given two distributions sampled by Boolean circuits C and D, $\Pi_{\text{Y es}} =$

$\{\langle C, D \rangle \text{ — dist}(C, D) \leq 1/n\}$ and $\Pi_{\text{No}} = \{\langle C, D \rangle \text{ — dist}(C, D) \geq 1 - 1/n\}$.

Here dist denotes the statistical distance between the distributions. When one of the distributions is the uniform distribution, the above problem is called Statistical Difference to Uniform (SDU). The seminal work of Sahai and Vadhan showed that SD is complete for the class SZK [24] and Goldreich, Sahai and Vadhan showed that SDU is complete for NISZK [16].

Entropy Approximation (EA) : Given a samplable distribution C and an integer k, $\Pi_{\text{Y es}} = \{\langle C, k \rangle \text{ — H}(C) \geq k + 1\}$ and $\Pi_{\text{No}} = \{\langle C, k \rangle \text{ — H}(C) \leq k - 1\}$, where H is the entropy function.

Goldreich, Sahai and Vadhan showed that Entropy Approximation is complete for NISZK. In the above problem, if the entropy function H is replaced with the min-entropy

function H_∞ , the corresponding problem is known as Min-entropy Approximation (MEA). Watson showed that MEA is complete for SBP [25]. It is interesting to note that while Entropy Approximation is NISZK-complete, the analogous Min-entropy Approximation problem is complete for SBP.

To establish our results, we study variants of the above distribution testing problems. The following problem, known as Uniform, is defined by Malka and was shown to be complete for NIPZK [21].

Uniform: Given a circuit $D : \{0, 1\}^m \rightarrow \{0, 1\}^{n+1}$, let $D[1 \dots n]$ denote the distribution of the first n bits of D and let $D[n+1]$ denote the distribution of the last bit of D . $\PiY_{es} = \{\langle D \rangle \mid D[1 \dots n] = U_n, \Pr[D[n+1] = 1] \geq 2/3\}$ and $\PiNo = \{\langle D \rangle \mid \text{sup}(D) \cap \{0, 1\}^n \leq 2n/3\}$.

Here U_n denotes the uniform distribution over n -bit strings and $\text{sup}(D)$ is the support of the distribution

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 , $\PiY_{es} = \{\langle D \rangle \mid D = U\}$ and $\PiNo = \{\langle D \rangle \mid \text{sup}(D) \leq 2n/2\}$

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 , $\PiY_{es} = \{\langle C, D \rangle \mid \text{sup}(C) \cap \text{sup}(D) = \emptyset\}$ and $\PiNo = \{\langle C, D \rangle \mid C = D\}$ (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 $PZK \subseteq PP$ 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 U_n 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 $C(U_m)$. 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 $\text{sup}(D)$ to denote the set of strings for which $D(x) \neq 0$.

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

$$\text{dist}(C, D) = \max(C(T) - D(T)) =$$

$$\begin{aligned} T &\subseteq S \\ (C(x) - D(x)) \\ C(x) &\not\subseteq D(x) \end{aligned}$$

Complexity Classes We refer the reader to the textbook by Arora and Barak [4] for definitions of standard 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 (\PiY_{es}, \PiNo) is said to belong to the complexity class SBP if there exists a constant $\epsilon > 0$, a polynomial $p(\cdot)$, and a probabilistic polynomial-time Turing Machine M such that

- If $x \in \PiY_{es}$ then $\Pr[M \text{ accepts}] \geq 1 + \epsilon$
- If $x \in \PiNo$ then $\Pr[M \text{ accepts}] \leq 1 - \epsilon$

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]). $MA \subseteq SBP \subseteq AM$ and $SBP \subseteq BPPpath \subseteq PP$.

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 $\langle P, V \rangle$, the prover and verifier. On input x and random strings rI, rP , P sends a message $\pi = P(x, rP, rI)$ to V , and

V computes $m = V(x, \pi, rI)$. V accepts x if $m = 1$, and rejects if $m = 0$. The transcript of the interaction is the tuple $\langle x, rI, \pi, m \rangle$.

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 (\PiY_{es}, \PiNo) is in NIPZK (Non-Interactive Perfect Zero Knowledge) if there is a non-interactive protocol $\langle P, V \rangle$ where V runs in polynomial time, and a randomized, polynomial-time computable simulator S , satisfying the following conditions:

- (Soundness:) For any function P and any $x \in \PiNo$, $\Pr[V \text{ accepts}] \leq 1/3$
- (Completeness:) If $x \in \PiY_{es}$, $\Pr[V \text{ accepts}] \geq 2/3$
- (Zero Knowledge:) For any $x \in \PiY_{es}$, the distribution of $S(x)$ is identical to the distribution of the transcript generated by $\langle P, V \rangle$ 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 transcript generated by $\langle P, V \rangle(x)$ 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: $\Pr_{x,y \sim D}(x = y)$. The following lemma is folklore. See [15] for a proof.

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

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 $T = \{x \in \{0, 1\}^n \mid x_1 \in \text{sup}(D)\}$. Suppose that

$|T| \leq 2n/3$ and $\Pr[D[n+1] = 1] = 1 + \epsilon$ for some $\epsilon \geq 0$. Then $\text{dist}(D[1 \dots n], \text{Un})$ is at least ϵ . Proof. Recall that $\text{dist}(D[1 \dots n], \text{Un}) = \max_{S \subseteq \{0, 1\}^n} \Pr_{d \sim D[1 \dots n]} [d \in S] - \Pr_{u \sim \text{Un}} [u \in S]$

$$\max_{S \subseteq \{0, 1\}^n} \Pr[d \in S] - \Pr[u \in S] \geq \Pr[d \in T] - \Pr[u \in T]$$

$$S \subseteq \{0, 1\}^n$$

$$d \sim D[1 \dots n]$$

$$u \sim \text{Un}$$

$$d \sim D[1 \dots n]$$

$$\geq 1 + \epsilon - \Pr$$

$$u \sim \text{Un}$$

$$[u \in T]$$

$$3 u \sim \text{Un}$$

$$3 2n$$

$$1 2n n$$

$$\geq + \epsilon - /2 = \epsilon$$

$$3 3$$

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

Proof. Recall the definition of Uniform: Given a circuit $D : \{0, 1\}^m \rightarrow \{0, 1\}^{n+1}$, ΠY es = $\{D : D[1 \dots n] = \text{Un}, \Pr[D[n+1] = 1] \geq 2/3\}$ and $\Pi No = \{D : \text{sup}(D) \cap \{0, 1\}^n \leq 2n/3\}$.

Consider the following randomized algorithm: Given D as input, get two samples d_0 and d_1 from D . If the first n bits of both d_0 and d_1 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 $D[1 \dots n]$ is at least 1 away from Un , or

$$1 6$$

$D[n+1] = 1$ 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 d_0 and d_1 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 ≥ 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 $\leq 1 + 3\log(37/36)$. For large enough n , $37 1 \geq (1 + 1)(1 + 3\log(37/36))$, so this is

$$2n 3n$$

a CoSBP algorithm 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 $\Pi = \langle \Pi Y, \Pi No \rangle$ be a promise problem whose instances are circuits. If there is an oracle circuit family $\{C_n\}_{n \geq 0}$ and a constant $c \geq 1$ with the following properties:

- C_n 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 $\{A_n\}_{n \geq 0}$, $\{B_n\}_{n \geq 0} \subseteq \{0, 1\}^{cn}$ such that for all n , $CAn \in \Pi Y$ and $CBn \in \Pi No$

$$n n$$

- For every probabilistic polynomial-time Turing Machine M and infinitely many n , for every $D_i \in \{A_i, B_i, \emptyset\}$, $1 \leq i \leq n$

$$(\cup_{n=1}^n D_i) \cup A_i$$

$$(\cup_{n=1}^n D_i) \cup B_i$$

$$\Pr[M$$

$$i=1 i$$

$$n (C_n i=1$$

$$n-1$$

$$) \text{accepts}$$

$$i 2,$$

$$\Pr[M (\cup_{n=1}^n D_i) \cup B_n (C(\cup_{i=1}^n D_i) \cup B_n) \text{accepts}]$$

$$i=1 n$$

then there exists an oracle O such that $\Pi O \notin \text{SBP}$

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 $p(\cdot)$ 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 $p(\cdot)$ the threshold polynomial for M .

Let $\{M_i\}_{i \geq 0}$ be an enumeration of the probabilistic polynomial-time machines. We consider an enumeration of tuples $\langle M_i, j \rangle_{i \geq 0, j \geq 0}$. In this enumeration considering $\langle M_i, j \rangle$ corresponds to the possibility that M_i is a SBP machine with threshold polynomial n_j . We first start with an empty oracle. Let $O_i = O \cap \{0, 1\}^{ci}$. For each i , O_i will be one of \emptyset , A_i or B_i . Consider $\langle M_i, j \rangle$ and let n be a length for which O_n is not yet defined and for which the inequality from the lemma holds for the machine M_i . Suppose that M_i makes queries of length $\leq m$. Note that by this, we have defined O_i for all $i \leq cn$, thus $O \subseteq \{0, 1\}^{cn}$ and for every $i \leq n$, O_i

is either \emptyset , A or B . Suppose that the acceptance probability of $M_i O \cup B_n$ (CAn) is

i i i

) is less than 2/2

. We set O

at length cn as A_n and for all the lengths from $cn + 1$ to m the oracle O is set to be \emptyset . Now C_{A_n} is a positive instance for which M_i cannot be a SBP machine with n_j as the threshold polynomial. Then we set O at length cn as A_n and move to the next tuple in the enumeration. Suppose that $M \cup A_n$ (C_{A_n}) accepts with

probability at least $2/2n_j$

n_j

. Now by the inequality from lemma 3, the acceptance probability of $M \cup B_n$ (C_{B_n})

B_n

is more than 1/2

. Note that C

is a negative instance for which M_i is not a SBP machine with threshold

polynomial n_j . Thus we make the oracle O at length cn to be B_n . It is easy to see that IIO is not in SBPO: Suppose not, and there exists a probabilistic polynomial-time machine M_i with threshold polynomial n_j . When we considered the tuple $\langle M_i, j \rangle$, we ensured that M_i does not have threshold polynomial n_j on C_{Ocn} .

- 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 SBP.

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 A_n and B_n 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 an 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 $f : [N] \rightarrow [N]$ 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 $1/2^k$. Thus the probability that the machine M accepts is $S \Pr[M \text{ accepts } S]$, summed over all sequences of size k.

Now given a subset D of $\{0, 1\}^n$ of size $2^{n/2}$, 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 $S \Pr[M \text{ accepts } S]$. 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 A_n and B_n is to encode the uniform distribution in A_n and the distribution UD in B_n . The set A_n can be defined as $\{\langle i, j \rangle \mid \text{the } i\text{th bit of the } j\text{th string of } \Sigma_n \text{ is } 1\}$ (in the standard lexicographical ordering). To define B_n given D, first consider the multiset D that contains $2^{n/2}$ copies of each elements of D. Thus the cardinality of D is 2^n . Now, the set B_n can be defined as tuples

$\langle l, j \rangle$ where the l th bit of the j th 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 $j \in \{1, \dots, 2^n\}$, $CO(j)$ outputs $O(\langle l, j \rangle)$ for all l between 1 and n. In other words, $CO(j)$ outputs the j th 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 $\langle l, j \rangle$ will encode the l th 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 $1 \leq i \leq 2n!$ index the set of all $2n!$ permutations of $\{0, 1\}^n$. Oracles for Yes instances: $Yi = \{\langle l, j \rangle : \text{the } l\text{th bit of the } j\text{th string of the } i\text{th permutation of } \{0, 1\}^n \text{ is } 1\}$.

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: $ND = \{\langle l, j \rangle : \text{the } l\text{th bit of the }$

i

j th string of the i th 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 $\{1, \dots, 2n!\}$ and D is uniformly chosen from all size $2m$ subsets of

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

n ,

$\Pr_{i,r}[AYi \text{ accepts } CYi]$

$\leq c$

$\Pr_i[\text{AND}_i \text{ accepts } CNDi]$

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 i th bit of the j th string, it evaluates $CO(j)$ and it extracts the i th 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 q_1, \dots, q_k to denote the circuit queries, and denote the output $CO(q_i)$ by ui . We can assume without loss of generality that all qi are distinct. We

use S to denote a typical tuple of answers $\langle u_1, \dots, u_k \rangle$. 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

$$ui = uj, i \neq j.$$

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 $\Pr[A \text{ accepts a Yes instance}]$, and it will reduce $\Pr[A \text{ accepts a No instance}]$. 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 $\geq c$.

We will use the following notation.

- “ AO asks $\langle q, i \rangle$ ” is the event that “the i th circuit query made by A is $CO(q)$.” For simplicity, we write this event as “ AO asks qi .”
- “ AO gets $\langle u, i \rangle$ ” is the event that “ $CO(q) = u$ where q is the i th query”. Again, for simplicity, we write this event as “ AO gets ui .”
- For $S = \langle u_1, \dots, u_k \rangle$, “ AO gets S ” is the event that “ AO gets u_1 and AO gets u_2 and \dots AO gets u_k (in that order)”.

Lemma 5. For any probabilistic algorithm A and for any fixed $S = \langle u_1, \dots, u_k \rangle$ where all ui are distinct,

$$\Pr[AYi \text{ gets } S \text{ and accepts}] = \Pr[A \text{ accepts}] \cdot Y 1$$

Proof.

$$\Pr[AYi \text{ gets } S \text{ and accepts}] = \Pr[AYi \text{ gets } S] \times \Pr[AYi \text{ accepts} \mid AYi \text{ gets } S]$$

i,r

i,r

i,r

$$= \Pr[AYi \text{ gets } S] \times \Pr[AS \text{ accepts}]$$

$i,r r$

The last equality is because AS is independent of i as discussed before. We will show that $\Pr_{i,r}[AYi \text{ gets } S] =$

$$Qk-1 \cdot 1$$

$k-1$

$$\Pr[AYi \text{ gets } S] = \Pr[AYi \text{ gets } u_{j+1} \mid AYi \text{ gets } \langle u_1, u_2, \dots, u_j \rangle]$$

For any fixed j let Ej denote the event “ AYi gets $\langle u_1, u_2, \dots, u_j \rangle$ ”. Then,

$$\Pr[AYi \text{ gets } u_{j+1} \mid AYi \text{ gets } \langle u_1, u_2, \dots, u_j \rangle] = \Pr[AYi \text{ gets } u_{j+1} \mid Ej]$$

i,r

i,r

$$= \sum_{q=j+1}^{k-1} \Pr[AYi \text{ asks } q \mid Ej] \times \Pr[CYi(q) = u_{j+1} \mid Ej]$$

$$= \sum_{q=j+1}^{k-1} \Pr[AYi \text{ asks } q \mid Ej] \times \Pr[CYi(q) = u_{j+1} \mid Ej]$$

$$= \sum_{q=j+1}^{k-1} \Pr[AYi \text{ asks } q \mid E] \times 1$$

$$= 1 \times \sum \Pr[\text{AY}_i \text{ asks } q \rightarrow E]$$

$$(2n - j)$$

$$1$$

$$= (2n - j)$$

$$qj+1$$

$$i,r$$

$$j+1 \ j$$

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 u_{j+1} . The lemma follows.

Lemma 6. For any algorithm A and any fixed $S = \langle u_1, \dots, u_k \rangle$ where u_i s are distinct,

$$k-1 \ m$$

$$n-m$$

$$\Pr[\text{AND}_i \text{ gets } S \text{ and accepts}] = \Pr[A \\ \text{accepts}] \times Y(2-j)2$$

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

$$\Pr[\text{AND}_i \text{ gets } S \text{ and accepts}] = \Pr[\text{AND}_i \text{ gets } S] \times \Pr[\text{AND}_i \text{ accepts} \rightarrow \text{AND}_i \text{ gets } S]$$

$$i,r,D$$

$$i,r,D$$

$$i,r,D$$

$$= \Pr[\text{AND}_i \text{ gets } S] \times \Pr[\text{AS accepts}]$$

$$i,r,D \ r$$

The last equality is because AS is independent of i and D.

We will show that $\Pr_{i,r,D}[\text{AND}_i \text{ gets } S] =$

$$Q^{k-1} (2m-j)2^{n-m}$$

$$k-1$$

$$\Pr[\text{AND}_i \text{ gets } S] = \Pr[\text{AND}_i \text{ gets } u_{j+1} \rightarrow \text{AND}_i \text{ gets } \langle u_1, u_2, \dots, u_j \rangle]$$

We will reuse the notation E_j for convenience. For any fixed j, let E_j denote the event “ $\text{AND}_i \text{ gets } \langle u_1, u_2, \dots, u_j \rangle$ ”. Then,

$$\Pr[\text{AND}_i \text{ gets } u_{j+1} \rightarrow \text{AND}_i \text{ gets } \langle u_1, u_2, \dots, u_j \rangle] = \Pr[\text{AND}_i \text{ gets } u_{j+1} \rightarrow E_j]$$

$$i,r,D$$

$$i,r,D$$

$$B. = \Sigma$$

$$\Pr[\text{AND}_i \text{ asks } q_{j+1} \rightarrow E_j] \times \Pr[\text{CND}_i(q_{j+1}) = u_{j+1} \rightarrow E_j]$$

$$q_{j+1}$$

$$= \sum \Pr[\text{AND}_i \text{ asks } q_{j+1} \rightarrow E_j] \times \Pr[\text{CND}_i(q_{j+1}) = u_{j+1} \rightarrow E_j]$$

We will show that for any q, \Pr

$$[CY_i(q) = u]$$

$$\rightarrow E] = (2m-j)2^{n-m}.$$

$$i,D$$

$$j+1 \ j$$

$$(2n-j)2$$

$$\Pr[\text{CND}_i(q) = u_{j+1} \rightarrow E_j] = \Pr[u_{j+1} \in D \rightarrow E_j] \times \Pr[\text{CND}_i(q) = u_{j+1} \rightarrow u_{j+1} \in D, E_j]$$

$$i,D$$

$$i,D$$

$$2n-j-1$$

$$n-m$$

$$i,D$$

$$= 2n-j \ 2m-j$$

$$2m-j$$

$$2n-j$$

$$2n-m$$

$$= 2n-j \times 2n-j$$

$$(2m-j)2^{n-m}$$

$$= (2n-j)2$$

The second equality is because of the following reasoning.

There are $2n-j$ choices of D where $u \dots u_j$

are included, and $2n-j-1$ that include u_{j+1} as well. Given that $u_1, \dots, u_{j+1} \in D$, the probability that

$$n-m$$

$\text{CND}_i(q_{j+1}) = u_{j+1}$ is 2 (since there are $2n-m$ copies of u_{j+1} remaining, and $2n-j$ total things remaining).

We need the following claim.

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

$$k-1 \ n$$

$$2-j \leq c$$

$$2n-2n/2j$$

$$j=0$$

Proof.

$$k-1 \ n \ k-1 \ n$$

$$Y2-j \leq Y2$$

$$j=0$$

$$2n-2n/2j$$

$$j=0 \ k-1$$

$$2n-2n/2j$$

$$n/2$$

$$= 2$$

$$2n/2-j$$

$$j=0$$

$$2n/2-k$$

$$\leq 2n/2-k$$

$$k \ k$$

$$= 1 + 2n/2-k$$

For any polynomial $k = k(n)$, $\lim_{n \rightarrow \infty} (1 + k(n))k(n) = 1$.

Hence the claim.

We can now prove Lemma 4.

Proof (Proof of Lemma 4).

From lemmas 5 and 6, we have

$$\Pr_{i,r}[\text{AY}_i \text{ accepts } CY_i]$$

$$\sum \Pr_{i,r}[\text{AY}_i \text{ gets } S \text{ and accepts}]$$

$$C. = \Sigma$$

$\Pr_{i,r,D}[\text{AND}_i \text{ accepts } CND] \ \Pr_{i,r,D}[\text{AND}_i \text{ gets } S \text{ and accepts}]$

$$\sum \Pr_{i,r}[\text{AY}_i \text{ gets } S \text{ and accepts}]$$

$$\leq S \text{ distinct}$$

$$S \sum \Pr_{i,r,D}[\text{AND}_i \text{ gets } S \text{ and accepts}]$$

$$\sum \Pr[A \text{ accepts}] \times Q^{k-1}$$

$$\sum \Pr[A \text{ accepts}] \times Q^{k-1} (2m-j)2^{n-m}$$

$$k-1 \ n$$

$$= 2-j (\text{substituting } m = n/2) 2n - 2n/2j$$

j=0

$\vdash c$ (by Claim 4.1)

The second equality follows because when the oracle is Y_i , S is always disjoint (as we never ask the same query twice) and when the oracle is ND_i 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

$\Pr[A Y_{i^*} \text{ accepts } CY_{i^*}]$

ND^*

ND^*

$] \vdash c$

$\Pr[A i^* \text{ accepts } C i^*]$

for every $c \in \{0, 1\}$ (by Lemma 4). Now define A_n as Y_{i^*} and B_n as ND^* . This looks very close to the conditions of Lemma 3 except that we restricted the oracles to be A_n and B_n . However, for Lemma 3, we require that oracles are of the form $(\bigcup_{i=1}^{n-1} D_i \cup A_n)$ and $(\bigcup_{i=1}^{n-1} D_i \cup B_n)$. To establish this, we resort to the standard techniques

$i=1 \dots i=n$

used in oracle constructions. Observe that the sets A_n and B_n can be constructed in double exponential

$n_j - 1$

time. Let $n_1 = 2$ and $n_j = 2^{2j-2}$. We will satisfy the conditions of Lemma 3 at lengths of the form n_j . For

$n_j - 1$

$\bigcup_{i=1}^{n_j-1} D_i \cup A$

every i that is not of the form n_j , we set both A and B to empty. Now $M = \bigcup_{i=1}^{n_j-1} D_i \cup A_{n_j}$ ($C \vdash i \leq n_j$) can

$j \vdash i \leq n_j$

be simulated using M_{Anj} (C_{Anj}). As for queries whose length does not equal $c \cdot n_j$, 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 G_1 and G_2 be two graphs, and let C_i be the distribution obtained by randomly picking a permutation π and outputting $\pi(G_i)$. Observe that if G_1 and G_2 are not isomorphic then the supports of C_1 and C_2 are disjoint, and if G_1 is isomorphic to G_2 , then $C_1 = C_2$. Moreover the distributions C_1 and C_2 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 C_0 denote the circuit

obtained by fixing the first input bit of C to be 0, and the circuit C_1 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 C_0 and C_1 are disjoint” or “ C_0 and C_1 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 C_n and families of sets A_n and B_n that satisfy the conditions of Lemma 3.

Definition 6 (Oracle families). Let $i \in \{1, \dots, 2n\}$ index the partitions of $\{0, 1\}^n$ into two sets S_0 and

$i \vdash i$

Oracles for Yes instances: Y_{ijk} is an oracle for the set $\{\langle 0, 1, m \rangle : \text{the } l\text{th bit of the } m\text{th string in the } j\text{th}$

permutation of $S_0 = 1\} \cup \{\langle 1, 1, m \rangle : \text{the } l\text{th bit of the } m\text{th string in the } k\text{th permutation of } S_1 = 1\}$.

$i \vdash i$

Oracles for No instances: We construct the No instances similarly, except both 0 and 1 cases query S_0 . That is, N_{ijk} is an oracle for the set $\{\langle 0, 1, m \rangle : \text{the } l\text{th bit of the } m\text{th string in the } j\text{th permutation of }$

$S_0 = 1\} \cup \{\langle 1, 1, m \rangle : \text{the } l\text{th bit of the } m\text{th string in the } k\text{th permutation of } S_0 = 1\}$

$i \vdash i$

An oracle of the above form will be denoted by O which is a disjoint union of sets denoted by O_0 and

O_1 . Now we define the input circuits that sample the two distributions.

Definition 7 (Oracle circuits). Let C_O be a fixed linear-size oracle circuit, with $n+1$ inputs and n outputs, defined as follows: on input $\langle 0, j \rangle$ where $j \in \{1, \dots, 2n\}$, $C_O(j)$ outputs $O_0(\langle l, j \rangle)$ for all l between 1 and n , and on input $\langle 1, j \rangle$ where $j \in \{1, \dots, 2n\}$, $C_O(j)$ outputs $O_1(\langle l, j \rangle)$ for all l between 1 and n . In other words, $C_O(\langle 0, j \rangle)$ outputs the j th string of O_0 and $C_O(\langle 1, j \rangle)$ outputs the j th string of O_1 .

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 $\{1, \dots, 2n\}$, $\{1, \dots, 2n-1\}$, $\{1, \dots, 2n-1\}$ respectively, then for any probabilistic polynomial-time algorithm A , for any constant $c \in \{0, 1\}$, for large enough n ,

$\Pr[i, j, k \vdash A Y_{ijk} \text{ accepts } CY_{ijk}]$

$\leq c$

$\Pr[i, j, k \vdash A N_{ijk} \text{ accepts } CN_{ijk}]$

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, p_1, \dots, p_h are the queries that begin with 0 and u_i is the result of query p_i . q_1, \dots, q_h are the queries that begin with 1 and v_i is the result of query q_i . S is the ordered multiset $\{u_1, v_1, \dots, u_h, v_h\}$.

$vh\rangle$. Notice that this is without loss of generality as A can simulate the original algorithm by ignoring either q_i or p_i as appropriate. The second difference is that, instead of assuming A rejects if any u_i matches any u_j , we assume A rejects if any u_i matches any v_j .

Lemma 8. For any probabilistic algorithm A and for any fixed $S = \langle u_1, v_1, \dots, u_h, v_h \rangle$ where all elements of

S are distinct,

$$\Pr [AY_{ijk} \text{ gets } S \text{ and accepts}] = \Pr[A \text{ accepts}] \times Y 1$$

Proof. Note that

Pr

i,j,k,r

$$[AY_{ijk} \text{ gets } S \text{ and accepts}] = \Pr$$

i,j,k,r

$$[AY_{ijk} \text{ gets } S] \times \Pr$$

i,j,k,r

$$[AY_{ijk} \text{ accepts} \rightarrow AY_{ijk} \text{ gets } S]$$

$$= \Pr [AY_{ijk} \text{ gets } S] \times \Pr [AY_{ijk} \text{ accepts}]$$

i,j,k,r r S

Thus we need to prove that

Pr

i,j,k,r

h-1

$$[AY_{ijk} \text{ gets } S] =$$

$$(2n - 2l)(2n - 2l - 1)$$

A=0

We use EA to denote the event AY_{ijk} gets $\langle u_1, v_1, \dots, u_A, v_A \rangle$. Note that

and

Pr

i,j,k,r

h-1

$$[AY_{ijk} \text{ gets } S] =$$

A=0

Pr

i,j,k,r

$$[AY_{ijk} \text{ gets } u_{A+1}, v_{A+1} \rightarrow EA]$$

Pr

i,j,k,r

$$[AY_{ijk} \text{ gets } u_{A+1}, v_{A+1} \rightarrow EA] =$$

p,q

Pr

i,j,k,r

$$[AY_{ijk} \text{ asks } p_{A+1} \text{ and } q_{A+1} \rightarrow EA]$$

x Pr

i,j,k,r

$$[CY_{ijk} (p) = u_{A+1} \text{ and } CY_{ijk} (q) = v_{A+1} \rightarrow AY_{ijk} \text{ asks } p_{A+1} \text{ and } q_{A+1}, EA]$$

i i i

$$\times \Pr[u_{A+1} \text{ is the } p\text{th element of } S_0 \rightarrow EA]$$

- A+1 i

$$\times \Pr[v_{A+1} \text{ is the } q\text{th element of } S_1 \rightarrow EA]$$

- 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

$$[AY_{ijk} \text{ gets } u_{A+1}, v_{A+1} \rightarrow EA] =$$

p,q

Pr

i,j,k,r

$$[AY_{ijk} \text{ asks } p_{A+1} \text{ and } q_{A+1} \rightarrow EA]$$

x Pr

i,j,k,r

$$[CY_{ijk} (p) = u_{A+1} \text{ and } CY_{ijk} (q) = v_{A+1} \rightarrow AY_{ijk} \text{ asks } p_{A+1} \text{ and } q_{A+1}, EA]$$

$$= 1 (2n - 2l)(2n - 2l - 1)$$

p,q

Pr

i,j,k,r

$$[AY_{ijk} \text{ asks } p$$

A+1

and q

A+1

→ EA]

1

$$= (2n - 2l)(2n - 2l - 1)$$

Since $\Pr_{i,j,k,r} [AY_{ijk} \text{ gets } S] = Q_{h-1} \Pr_{i,j,k,r} [AY_{ijk} \text{ gets } u_{A+1}, v_{A+1} \rightarrow EA]$, using this with the above derived equality we obtain that

Pr

i,j,k,r

h-1

$$[AY_{ijk} \text{ gets } S] =$$

$$(2n - 2l)(2n - 2l - 1)$$

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 $S = \langle u_1, v_1, \dots, u_h, v_h \rangle$ that are all distinct,

h-1 n n

$$\Pr [AN_{ijk} \text{ gets } S \text{ and accepts}] = \Pr[A \text{ accepts}] \times Y (2 - 2l)(2 - 2l - 1) 1$$

i,j,k,r

Proof. As before,

r S

A=0

$$(2n - 2l)(2n - 2l - 1) (2n - 1 - l) 2$$

Pr

i,j,k,r

$$[AN_{ijk} \text{ gets } S \text{ and accepts}] = \Pr$$

i,j,k,r

$$[AN_{ijk} \text{ gets } S] \times \Pr$$

i,j,k,r

$$\begin{aligned} & [\text{ANijk accepts} \rightarrow \text{ANijk gets } S] \\ &= \Pr[\text{ANijk gets } S] \times \Pr[\text{ANijk accepts}] \end{aligned}$$

It suffices to show that

$$i,j,k,r \in S$$

$$h-1 \leq n \leq$$

Pr

$$i,j,k,r$$

$$[\text{ANijk gets } S] = (2 - 2l)(2 - 2l - 1) 1$$

$$(2n - 1 - 2l)(2n - 1 - 2l - 1)(2n - 1 - 2l - 2)$$

$$A=0$$

If EA denotes the event “ANijk gets $\langle u_1, v_1, \langle \cdot, u_A, v_A \rangle \rangle$ ”, then

Now,

Pr

$$i,j,k,r$$

$$h-1$$

$$[\text{ANijk gets } S] = \Pr[\text{ANijk gets } u_A+1, v_A+1 \rightarrow \text{EA}]$$

$$ijk \in$$

$$A=0$$

$$\Pr[\text{ANijk gets } u_A+1, v_A+1 \rightarrow \text{EA}] = \sum \Pr[\text{ANijk asks } p_A+1 \text{ and } q_A+1 \rightarrow \text{EA}]$$

$$\times \Pr[\text{CNijk}(p) = u_A+1 \text{ and } \text{CNijk}(q) = v_A+1 \rightarrow \text{EA}, \text{ANijk asks } p_A+1 \text{ and } q_A+1]$$

$$ijk \in$$

Consider the event “CNijk(p) = u_A+1 and CNijk(q) = v_A+1”, conditioned on EA and “ANijk asks p_A+1 and q_A+1”. For this event to happen, it must be the case that both u_A+1 and v_A+1 are in S0, and u_A+1 is the pth

$$i A+1$$

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

$$i A+1 \in i$$

that EA and A asks p_A+1 and q_A+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 u_A+1 is the pth element given EA is $1/(2n - 1 - l)$ and similarly, the probability that

v_A+1 is the qth element given EA is $1/(2n - 1 - l)$.

Thus

$$\Pr[\text{AN} \rightarrow$$

$$ijk \in$$

$$ijk \text{ gets } u_A+1, v_A+1 \rightarrow \text{EA}] =$$

$$=$$

$$(2n - 1 - 2l)(2n - 1 - 2l - 1)(2n - 2l)(2n - 2l - 1)$$

$$(2n - 1 - 2l)(2n - 1 - 2l - 1)(2n - 2l)(2n - 1 - 2l - 1)$$

$$1(2n - 1 - l)2$$

$$1$$

$$(2n - 1)2$$

$$\Sigma p, q$$

$$\Pr[\text{AN} \rightarrow$$

$$ijk \in$$

$$ijk \text{ asks } p_A+1 \text{ and } q_A+1 \rightarrow \text{EA}]$$

Thus

$$h-1$$

$$n-1$$

$$n-1$$

$$\Pr[\text{ANijk gets } S] = Y(2 - 2l)(2 - 2l - 1) 1 ,$$

$$i,j,k,r$$

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 $c \in \mathbb{C}$,

1, for large enough n,

$$h-1$$

$$n-1 2$$

$$(2 - l) + c$$

$$(2n - 1 - 2l)(2n - 1 - 2l - 1)$$

$$A=0$$

Proof.

$$h-1$$

$$n-1 2$$

$$h-1$$

$$n-1 2$$

$$Y(2 - l) \leq Y(2 - 1)$$

$$A=0 A=0$$

$$h-1$$

$$\leq$$

$$A=0$$

$$1 + 1 2$$

$$1 + 2n - 1 - 2l - 1)$$

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 Computational 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 Conference, 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.