A Complete Problem for Statistical Zero Knowledge*

Amit Sahai† Salil Vadhan‡

October 25, 2000

Abstract

We present the first complete problem for SZK*, the class of (promise) problems possessing statistical zero-knowledge proofs (against an honest verifier). The problem, called* STATISTICAL DIFFERENCE*, is to decide whether two efficiently samplable distributions are either statistically close or far apart. This gives a new characterization of* SZK *that makes* no reference to interaction or zero knowledge.

We propose the use of complete problems to unify and extend the study of statistical zero knowledge. To this end, we examine several consequences of our Completeness Theorem and its proof, such as:

- *A way to make every (honest-verifier) statistical zero-knowledge proof very communication efficient, with the prover sending only one bit to the verifier (to achieve soundness error* $1/2$*).*
- *Simpler proofs of many of the previously known results about statistical zero knowledge, such as the Fortnow and Aiello–Håstad upper bounds on the complexity of* SZK *and Okamoto's result that* SZK *is closed under complement.*
- *Strong closure properties of* SZK *which amount to constructing statistical zero-knowledge proofs for complex assertions built out of simpler assertions already shown to be in* SZK*.*
- *New results about the various measures of "knowledge complexity," including a collapse in the hierarchy corresponding to knowledge complexity in the "hint" sense.*
- *Algorithms for manipulating the statistical difference between efficiently samplable distributions, including transformations which "polarize" and "reverse" the statistical relationship between a pair of distributions.*

---

*Preliminary versions of this work appeared in the proceedings of the *38th Annual IEEE Symposium on the Foundations of Computer Science* [SV97] and the *DIMACS Workshop on Randomization Methods in Algorithm Design* [SV99].

†Department of Computer Science, Princeton University, Princeton, NJ 08544. Email: `sahai@cs.princeton.edu`. URL: `http://d.n.....w..r...uw.nd.....c.salvatore.rest/~sahai`. This work was done when the author was at the MIT Laborat
Computer Science, supported by a DOD NDSEG Graduate Fellowship and partially by DARPA grant DABT63-96-C-0018.

‡Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138. Email: `salil@deas.harvard.edu`. URL: `http://....ah....naae.wr..salvatore.rest/~salil`. This work was done when the author was
the MIT Department of Mathematics, supported by a DOD NDSEG Graduate Fellowship and partially by DARPA grant DABT63-96-C-0018.

Contents

1 Introduction

A revolution in theoretical computer science occurred when it was discovered that NP has complete problems [Coo71, Lev73, Kar72]. Most often, these theorems and other completeness results are viewed as negative statements, as they provide evidence of a problem's intractability. These same results, viewed as positive statements, enable one to study an entire class of problems by focusing on a single problem. For example, all languages in NP were shown to have computational zero-knowledge proofs when such a proof was exhibited for Graph 3-colorability [GMW91]. Similarly, the result that IP = PSPACE was shown by giving an interactive proof for Quantified Boolean Formula, which is complete for PSPACE [LFKN92, Sha92]. More recently, the celebrated PCP theorem characterizing NP was proven by designing efficient probabilistically checkable proofs for a specific NP-complete language [ALM$^+$98, AS98].

In this paper, we present a complete problem for SZK, the class of promise problems[1] possessing statistical zero-knowledge proofs (against an honest verifier). This problem provides a new and simple characterization of SZK — one which makes no reference to interaction or zero knowledge. We propose the use of complete problems as a tool to unify and extend the study of statistical zero knowledge. To this end, we use our complete problem to both establish a number of new results about SZK and easily deduce nearly all previous results about SZK.

1.1 Statistical zero knowledge

Zero knowledge was introduced in the seminal paper of Goldwasser, Micali, and Rackoff [GMR89] within the context of their new notion of interactive proof systems. Informally, an *interactive proof* is a protocol in which a computationally unbounded prover P attempts to convince a probabilistic polynomial-time verifier V of an assertion, namely that a string x is a YES instance of some (promise) problem. The *zero knowledge* property requires that, during this process, the verifier learns nothing beyond the validity of the assertion being proven! To formalize this seemingly impossible notion, two probability distributions are considered:

1. The interaction of P and V from V's point of view.
2. The output of a probabilistic polynomial-time machine not interacting with anyone, called the *simulator*, on input x.

An interactive proof system (P, V) is said to be *zero knowledge* if, for every YES instance x, the two distributions above are "alike." Intuitively, the verifier gains no knowledge by interacting with the prover except that x is a YES instance, since it could have run the simulator instead. The specific variants of zero knowledge differ by the interpretation given to "alike." The most strict interpretation, leading to *perfect zero knowledge*, requires that the distributions be identical. A slightly relaxed interpretation, leading to *statistical zero knowledge* (sometimes called *almost perfect zero knowledge*), requires that the distributions have negligible statistical difference from one another. The most liberal interpretation, leading to *computational zero knowledge*, requires that samples from the two distributions be indistinguishable by any polynomial-time machine.

In this work, we focus on the class of problems possessing *statistical* zero-knowledge proof systems, which we denote SZK. We remark that we are restricting our attention to zero-knowledge proofs against an *honest verifier*, *i.e.* the verifier that follows the specified protocol. In cryptographic

[1] A *promise problem* is a decision problem given by a pair of disjoint sets of strings, corresponding to YES and NO instances. In contrast to languages, there may be strings which are neither YES instances nor NO instances. A formal definition is given in Section 2.

applications, one usually wants the zero-knowledge condition to hold for all (even cheating) verifier strategies. However, subsequent to this work, it has been shown that one can transform any proof system that is statistical zero knowledge against an honest verifier into one that is statistical zero knowledge against all verifiers [GSV98], so restricting to honest verifiers causes no loss of generality.

SZK contains a number of important problems, including Graph Nonisomorphism [GMW91], a problem which is not known to be in NP. It also contains problems with cryptographic application and significance that are believed to be hard on average, such as Quadratic Residuosity (and its complement) [GMR89], a problem equivalent to the Discrete Logarithm problem [GK93], and approximate versions of the Shortest Vector and Closest Vector problems in lattices [GG00]. At the same time, the statistical zero knowledge property has several strong consequences. Unlike a computational zero-knowledge protocol, a statistical zero-knowledge protocol remains zero knowledge even against a computationally unbounded verifier. In addition, a problem which has a statistical zero-knowledge proof must lie low in the polynomial-time hierarchy. In fact, such a problem cannot be NP-complete unless the polynomial-time hierarchy collapses [For89, AH91, BHZ87]. Because SZK contains problems believed to be hard yet cannot contain NP-complete problems, it holds an intriguing position in complexity theory.

1.2 The complete problem

The promise problem we show to be complete for SZK is Statistical Difference. An instance of Statistical Difference consists of a pair of probability distributions, specified by circuits which sample from them. Roughly speaking, the problem is to decide whether the distributions defined by the two circuits are statistically close or far apart. (The gap between 'close' and 'far apart' is what makes it a promise problem and not just a language.) Our main theorem is that Statistical Difference is complete for SZK. This Completeness Theorem gives a new characterization of SZK. Informally, it says that the assertions that can be proven in statistical zero knowledge are exactly those that can be cast as deciding whether a pair of efficiently samplable distributions are statistically close or far apart.

The starting point for our proof of the Completeness Theorem is a powerful theorem of Okamoto [Oka00], which states that all languages in SZK have *public-coin* (also known as Arthur-Merlin [BM88]) statistical zero-knowledge proofs. Using the approach pioneered by Fortnow [For89], we analyze the simulator of such a proof system and show that statistical properties of the simulator's output distribution can be used to distinguish between YES and NO instances of the problem in consideration. Our key new observation is that, for a *public-coin* proof system, these statistical properties can be captured by the statistical difference between efficiently samplable distributions. We thereby conclude that every problem in SZK reduces to Statistical Difference.

To show that Statistical Difference is in SZK, we exhibit a simple 2-message proof system for it, generalizing the well-known proof systems for Quadratic Nonresiduosity [GMR89] and Graph Nonisomorphism [GMW91]. One ingredient in our proof system is a new "Polarization Lemma" for statistical difference, which may be of independent interest. Roughly speaking, this lemma gives an efficient transformation which takes as input a pair of probability distributions (specified by circuits which sample from them) and produces a new pair of distributions such that if the original pair is statistically close (resp., far apart), the new pair is statistically much closer (resp., much further apart).

1.3 Consequences

We propose using complete problems, such as Statistical Difference, to unify and extend the study of SZK. We also use the connection between SZK and statistical properties of samplable distributions to establish new techniques for manipulating such distributions. The results we obtain along these lines are summarized below.

The relationship between SZK and BPP. Our complete problem illustrates that statistical zero knowledge is a natural generalization of BPP. In the definition of Statistical Difference , the circuits can output strings of any length. If we restrict the circuits to have output of logarithmic length, the resulting problem is easily shown to be complete for BPP.

Efficient SZK proof systems. The zero-knowledge proof system we exhibit for Statistical Difference has many attractive properties (which we describe shortly); by the Completeness Theorem it follows that every problem in SZK also has a proof system with such properties. First, the protocol is very communication efficient — only two messages are exchanged between the prover and verifier, and the prover only sends *one bit* to the verifier (to achieve soundness error 1/2). In addition, we will show that when the input is a yes instance, the verifier's view of the interaction can be simulated by a polynomial-time simulator with *exponentially small* statistical deviation. Moreover, we will show that this simulator deviation can be made to shrink exponentially fast as a function of a separate "security parameter" which can be varied independently from the assertion being proven. This is in contrast to the definition of SZK, which only requires that the verifier be able simulate the interaction with statistical deviation $1/n^{\omega(1)}$, where n is the the length of the assertion being proven.

Closure properties. Using the complete problem, we demonstrate that SZK has some very strong closure properties. These can be informally described as asserting the existence of statistical zero-knowledge proofs for complex assertions built out simpler assertions already known to be in SZK. These complex assertions take the form of arbitrary propositional formulae whose atoms are statements about membership in some problem in SZK, and the statistical zero-knowledge proofs we exhibit have complexity which is polynomial in the size of these formulae. These results generalize earlier ones of De Santis, Di Crescenzo, Persiano, and Yung [DDPY94] and Damgård and Cramer [DC96], which held for monotone formulae and various subclasses of SZK, such as random self-reducible problems.

By the Completeness Theorem, the closure properties we establish are equivalent to the existence of efficient transformations that manipulate the statistical difference between samplable distributions in various ways. Indeed, it is by exhibiting such transformations that we prove the closure properties of SZK. The transformations we give (and their application to closure properties) are inspired by the techniques of De Santis, Di Crescenzo, Persiano, and Yung [DDPY94].

Simpler proofs of previous results. Many of the previous results about SZK can be deduced as immediate corollaries of our Completeness Theorem and its proof. For example, the result of Okamoto [Oka00] that SZK is closed under complement follows directly from our proof of the Completeness Theorem. Then, using the fact that our proof system for Statistical Difference is a constant-round one, we deduce that $\mathsf{SZK} \subset \mathsf{AM} \cap \mathsf{co\text{-}AM}$, as originally proven by Fortnow [For89] and Aiello and Håstad [AH91]. In addition, the result of Ostrovsky [Ost91] that one-way functions exist if SZK contains a hard-on-average problem follows immediately by combining our Completeness Theorem with a result of Goldreich [Gol90] on computational indistinguishability.

Knowledge complexity. In addition to introducing zero-knowledge proofs, the conference version of the paper of Goldwasser, Micali, and Rackoff [GMR89] proposed a more general idea of measuring the amount of knowledge leaked in an interactive proof. Goldreich and Petrank [GP99] suggested several definitions of *knowledge complexity* to accomplish this, and relationships between these various types of knowledge complexity were explored in [GP99, BP92, GOP98, ABV95, PT96]. Loosely speaking, the definitions of (statistical) knowledge complexity measure the "amount of help" a verifier needs to generate a distribution that is statistically close to its real interaction with the prover. There are several ways of formalizing the "amount of help" the verifier needs and each leads to a different notion of knowledge complexity.

Our work on SZK turns out to have consequences for (non-zero) knowledge complexity as well. First, we show that for the weakest of the various measures of knowledge complexity, namely statistical knowledge complexity in the "hint sense", the corresponding hierarchy collapses by logarithmic additive factors at all levels, and in particular, knowledge complexity $\log n$ equals statistical zero knowledge. No collapse was previously known for any of the variants of knowledge complexity suggested in [GP99]. Our results are obtained by combining our results on SZK with a general lemma relating knowledge complexity in the hint sense to zero knowledge *for promise problems.*

As with zero knowledge, *perfect* knowledge complexity can also be defined. This measures the number of bits of help the verifier needs to simulate the interaction *exactly*, rather than statistically closely. Using our complete problem for SZK, we improve some results of Aiello, Bellare, and Venkatesan [ABV95] on the perfect knowledge complexity of statistical zero knowledge.

Reversing statistical difference. One interesting result that follows from the completeness of Statistical Difference and the closure of SZK under complement is the existence of an efficient mapping which "reverses" statistical difference. That is, for every pair of efficiently samplable distributions, we can construct another pair of efficiently samplable distributions such that when the former are statistically close, the latter are statistically far apart, and when the former are far apart, the latter are close.

This motivated us to search for a more explicit description of such a transformation. By extracting ideas from the work of Okamoto [Oka00] and our proof of the Completeness Theorem, we have obtained such a description (which we give in Section 4.4).

Weak SZK and expected polynomial-time simulators. The original definition of SZK in [GMR89] allows the simulator to run in *expected* polynomial time, whereas we insist on strict polynomial time, following [Gol95]. Actually, our proof of the Completeness Theorem shows that the two definitions are equivalent for *public-coin* proof systems. That is, if a problem possesses a public-coin SZK proof system with an expected polynomial-time simulator, then it also possesses an SZK proof system with a strict polynomial-time simulator (which can be made public coin by [Oka00]). In fact, the equivalence extends to an even weaker definition of SZK, in which it is only required that for every polynomial $p(n)$, there exists a simulator achieving simulator deviation $1/p(n)$.

Perfect and computational zero knowledge. Our techniques can also be used to analyze public-coin perfect and computational zero-knowledge proofs. Although we do not obtain complete problems in these cases, we do obtain some novel insights into the corresponding complexity classes. Specifically, in Section 4.6 we show that every problem possessing a public-coin perfect zero-knowledge proof (essentially) reduces to a restricted version of STATISTICAL DIFFERENCE. We also show that for any problem possessing a public-coin computational zero-knowledge proof, there exist ensembles of samplable distributions indexed by instances of the problem such that on YES instances, the distributions are computationally indistinguishable and on NO instances, the distributions are statistically far apart.

Cheating-verifier zero knowledge. While in this paper we primarily focus on honest-verifier statistical zero knowledge, there have been a number of works examining "cheating-verifier" statistical zero knowledge, and in particular relating the honest and cheating-verifier definitions. Some of these works exhibited transformations from honest-verifier SZK proofs to cheating-verifier ones under (successively weaker) complexity assumptions ([BMO90], [OVY93], and [DGOW95, Part 2]), and others gave unconditional transformations for restricted subclasses of SZK ([Dam93] and [DGOW95, Part 1]). Finally, subsequent to our paper, it was proven in [GSV98] that honest-verifier and cheating-verifier SZK are the equal, unconditionally and with no restrictions.

Following the paradigm advocated in [BMO90], we use the above transformations to translate our results about honest-verifier SZK, namely the Completeness Theorem and its corollaries, to the cheating-verifier class. In Section 5, we precisely state the results thereby obtained for cheating-verifier statistical zero knowledge.

1.4 Subsequent work

Subsequent to the conference version of this paper [SV97], there have a been a number of other works improving our understanding of SZK, many of which make use of the complete problem methodology advocated here. As mentioned above, Goldreich, Sahai, and Vadhan [GSV98] show that honest-verifier statistical zero knowledge equals cheating-verifier statistical zero knowledge. Goldreich and Vadhan [GV99] use the complete problem methodology to give a simpler proof of Okamoto's theorem that private-coin SZK equals public-coin SZK (on which our work relies). In the process, they exhibit another complete problem for SZK, called ENTROPY DIFFERENCE, which amounts to deciding which of two given distributions (specified by circuits which sample from them) has noticeably higher entropy than the other. Di Crescenzo, Sakurai, and Yung [DSY00] consider two variants of (honest-verifier) statistical zero-knowledge proofs, namely "proofs of decision power" and "proofs of decision", and exhibit such proof systems for all of SZK. Their construction makes use of the complete problems for SZK given here and in [GV99] and special properties of their proof systems.

De Santis, Di Crescenzo, Persiano and Yung [DDPY98] extend the use of complete problems to study "noninteractive" statistical zero knowledge; they exhibit a complete problem for the corresponding complexity class NISZK and use it to prove some general results about the class. Goldreich, Sahai, and Vadhan [GSV99] exhibit two more complete problems for NISZK. These problems are natural restrictions of the complete problems for SZK given here and in [GV99], and thus they are able to use the complete problems to relate SZK and NISZK. Gutfreund and Ben-Or [GB00] examine weaker models of noninteractive zero knowledge proofs, and, using our complete problem and reversal mapping, show that every problem in SZK has a noninteractive statistical zero-knowledge proof in one of their models.

Finally, Vadhan [Vad00] examines the blow-up in the prover's complexity incurred by transformations from private-coin proof systems to public-coin proof systems, such as those in [GS89, Oka00], and shows that this inefficiency is inherent in the fact that the transformations use the original prover and verifier strategies as "black boxes". In fact, it is shown that any black-box transformation which preserves the prover's complexity must fail on our proof system for STATISTICAL DIFFERENCE.

Unified presentations of many of the above results, together with the results in this paper, can be found in the Ph.D. theses of the authors [Vad99, Sah00].

2 Preliminaries

2.1 Promise problems

The problem we prove to be complete for SZK is not a language, but rather a *promise problem* [ESY84]. Formally, a promise problem Π consists of two disjoint sets of strings Π_Y and Π_N, where Π_Y is the set of YES *instances* and Π_N is the set of NO *instances*. A promise problem Π is associated with the following computational problem: Given an input which is "promised" to lie in $\Pi_Y \cup \Pi_N$, decide whether it comes from Π_Y or Π_N. The *complement* of Π is the promise problem $\overline{\Pi}$, where $\overline{\Pi}_Y = \Pi_N$ and $\overline{\Pi}_N = \Pi_Y$. Note that languages are a special case of promise problems.

We say that promise problem Π *reduces* to promise problem Γ if there is a polynomial-time computable function f such that

$$\begin{aligned} x \in \Pi_Y &\Rightarrow f(x) \in \Gamma_Y \\ x \in \Pi_N &\Rightarrow f(x) \in \Gamma_N \end{aligned}$$

If C is a class of promise problems, we say that promise problem Π is *complete* for C if $\Pi \in \mathsf{C}$ and every promise problem in C reduces to Π. As above, all reductions we consider are polynomial-time many-one (or Karp) reductions, unless otherwise specified.

2.2 Probability distributions

If X is a probability distribution (or random variable), we write $x \leftarrow X$ to indicate that x is a sample taken from X. If S is a set, we write $x \in_R S$ to indicate that x is uniformly selected from S.

In this paper, we will consider probability distributions defined both by circuits and probabilistic algorithms (*i.e.* Turing machines). If A is a probabilistic algorithm, we use $A(x)$ to denote the output distribution of A on input x. A *PPT* algorithm (for "probabilistic polynomial time") is a probabilistic algorithm which runs in strict polynomial time. If C is a circuit mapping m-bit strings to n-bit strings, then choosing an input u uniformly at random from $\{0,1\}^m$ defines a probability distribution on $\{0,1\}^n$ given by $C(u)$. For notational convenience, we also denote this probability distribution by C. These definitions capture the idea of an "(efficiently) samplable" distribution, as to sample from the distribution one need only run the algorithm or evaluate the circuit.

2.3 The statistical difference metric

For probability distributions (or random variables) X and Y on a discrete set D, the *statistical difference* between X and Y is defined to be

$$\|X - Y\| = \max_{S \subset D} |\Pr[X \in S] - \Pr[Y \in S]|. \tag{1}$$

This is often also called the *variation distance* between X and Y. Removing the absolute values in (1) does not change the definition because replacing S by its complement changes the sign (but not magnitude) of $\Pr[X \in S] - \Pr[Y \in S]$. The maximum in (1) can be achieved by taking $S = \{x : \Pr[X = x] > \Pr[Y = x]\}$ (or its complement); this can be seen directly or in the proof of Fact 2.1 below.

There is an equivalent formulation of statistical difference in terms of the ℓ_1 norm $|\cdot|_1$ that will sometimes be more convenient for us. To every probability distribution X on a discrete set D, the *mass function* of X is a vector in $\mathbb{R}^D$ whose x'th coordinate is $\Pr[X = x]$. For the sake of elegance, we also denote this vector by X. With this notation, we can state the following well-known fact.

Fact 2.1 $\|X - Y\| = \frac{1}{2}|X - Y|_1$

The proof of this fact and others in this section are deferred to Appendix A. It is immediate from this characterization of statistical difference that it is a metric (as long as we identify random variables that are identically distributed). In particular, it satisfies the Triangle Inequality.

Fact 2.2 (Triangle Inequality) *For any probability distributions X, Y, and Z,*

$$\|X - Y\| \leq \|X - Z\| + \|Z - Y\|$$

Recall that for any two vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$, their *tensor product* $v \otimes w$ is the vector in $\mathbb{R}^{nm}$, whose (i, j)'th component is $v_i w_j$. Now, if we have a pair of random variables (X, Y) (on the same probability space) taking values in $D \times E$, then X is independent from Y iff the mass function of (X, Y) is the tensor product of the mass functions of X and Y (which are elements of $\mathbb{R}^D$ and $\mathbb{R}^E$, respectively). For this reason, if we have random variables X and Y taking values in sets D and E, respectively, we write $X \otimes Y$ for the random variable taking values in $D \times E$ which consists of independent samples of X and Y. Similarly, $\otimes^k X$ denotes the random variable taking values in D^k consisting of k independent copies of X, *i.e.* $X \otimes X \otimes \cdots \otimes X$.

Now, for any two vectors v and w, $|v \otimes w|_1 = |v|_1 \cdot |w|_1$. In addition, for any mass function X, $|X|_1 = 1$. These facts enable one to show that the statistical difference behaves well with respect to independent random variables:

Fact 2.3 *Suppose X_1 and X_2 are independent random variables on one probability space and Y_1 and Y_2 are independent random variables on another probability space. Then,*

$$\|(X_1, X_2) - (Y_1, Y_2)\| \leq \|X_1 - Y_1\| + \|X_2 - Y_2\|$$

One basic fact about statistical difference is that it cannot be created out of nothing. That is, for any procedure A, even if it is randomized, the statistical difference between $A(X)$ and $A(Y)$ is no greater than the statistical difference between X and Y. Formally, if D is any set, a *randomized procedure* on D is a a pair $A = (f, R)$, where R is a probability distribution on some set E and f is a function from $D \times E$ to any set F. Think of the distribution R as providing a "random seed" to the procedure A. If X is a probability distribution on D, then $A(X)$ denotes the probability distribution on F obtained by sampling $X \otimes R$ and applying f to the result. Note that applying a *function* is a special case of applying a randomized procedure.

Fact 2.4 *If X and Y are random variables and A is any randomized procedure, then*

$$\|A(X) - A(Y)\| \leq \|X - Y\|$$

The next fact is useful when arguing that the statistical difference between distributions is small.

Fact 2.5 *Suppose $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ are probability distributions on a set $D \times E$ such that*

1. *X_1 and Y_1 are identically distributed, and*

2. *With probability greater than $(1-\epsilon)$ over $x \leftarrow X_1$ (equivalently, $x \leftarrow Y_1$),*

$$\|X_2|_{X_1=x} - Y_2|_{Y_1=x}\| < \delta$$

(where $B|_{A=a}$ denotes the conditional distribution of B given that $A = a$ for jointly distributed random variables A and B).

Then $\|X - Y\| < \epsilon + \delta$.

The next fact says that if two distributions have small statistical difference, then their mass functions must be close at most points.

Fact 2.6 *If X and Y are any two distributions such that $\|X - Y\| < \epsilon$, then with probability $> 1 - 2\sqrt{\epsilon}$ over $x \leftarrow X$,*

$$\left(1 - \sqrt{\epsilon}\right) \Pr\left[X = x\right] < \Pr\left[Y = x\right] < \left(1 + \sqrt{\epsilon}\right) \Pr\left[X = x\right]$$

2.4 Zero-knowledge proofs

Before defining zero knowledge, we need to introduce some more terminology. Recall that a *PPT* algorithm is a probabilistic algorithm which runs in *strict* polynomial time. A function $f(n)$ is *negligible* if for all polynomials $p(n)$, $f(n) \le \frac{1}{p(n)}$ for all sufficiently large n.

We follow [GMR89] and [Gol95] in defining interactive proofs and zero-knowledge. The original definitions in [GMR89] were given for languages. We generalize these definitions to promise problems in the natural way, as previously done in [GK93]. That is, conditions previously required for inputs in the language are now required for YES instances of a promise problem and conditions previously required for inputs not in the language are now required for NO instances.

Informally, an interactive proof is a protocol in which a computationally unbounded prover attempts to convince a polynomial-time verifier V that an assertion is true, *i.e.* that a string x is a YES instance of a promise problem. More formally, an interactive protocol (P, V) between a computationally unbounded prover P and a PPT verifier V is said to be an *interactive proof system* for a promise problem Π with *completeness error* $c(n)$ and *soundness error* $s(n)$ if

1. If $x \in \Pi_Y$, then $\Pr\left[(P, V)(x) = \texttt{accept}\right] \ge 1 - c(|x|)$.

2. If $x \in \Pi_N$, then for all P^*, $\Pr\left[(P^*, V)(x) = \texttt{accept}\right] \le s(|x|)$.

We always require that $1 - c(n) > s(n) + 1/\mathrm{poly}(n)$ and that both can be computed in time $\mathrm{poly}(n)$; under this assumption, parallel repetition can be used to obtain a new interactive proof for Π with completeness error and soundness error 2^{-n^k}, for any constant k. We say that (P, V) *exchanges at $m(n)$ messages* if the prover and verifier exchange at most $m(n)$ messages on any input of length n. An interactive proof system is said to be *public coin* if on every input, the verifier's random coins r can be written as a concatenation of strings $r_1 r_2 \cdots r_l$ such that the i'th message sent from the verifier to the prover is simply r_i.

Roughly speaking, an interactive proof is said to be zero knowledge if, when the input is a YES instance, the verifier can simulate its view of the interaction on its own. To formalize this, let (P,V) be an interactive proof system (P,V) for a promise problem Π. Let $\mathrm{View}_{P,V}(x)$ be a random variable describing the random coins of V and the messages exchanged between P and V during their interaction on input x. (P,V) is said to be a *statistical zero-knowledge* proof system (against the honest verifier) if there exists a PPT simulator S and a negligible function α (called the *simulator deviation*) such that

$$\text{If } x \in \Pi_Y, \text{ then } \|S(x) - \mathrm{View}_{P,V}(x)\| \leq \alpha(|x|). \tag{2}$$

A *perfect zero-knowledge* proof system is defined in the same way, except that (2) is replaced by $\|S(x) - \mathrm{View}_{P,V}(x)\| = 0$, where S is allowed to output '`fail`' with probability at most 1/2 and $S(x)$ denotes the conditional distribution of S given that the output is not `fail`.[2] A *computational zero-knowledge* proof system replaces (2) with the requirement that $\{S(x)\}_{x\in\Pi_Y}$ and $\{\mathrm{View}_{P,V}(x)\}_{x\in\Pi_Y}$ are *computationally indistinguishable* [GM84, Yao82] ensembles of distributions. That is, for every *nonuniform* polynomial-time algorithm D, there is a negligible function α such that $|\Pr[D(x, S(x)) = 1] - \Pr[D(x, \mathrm{View}_{P,V}(x))]| \leq \alpha(|x|)$ for all $x \in \Pi_Y$.

We let SZK (resp. PZK, CZK) denote the class of promise problems with statistical (resp. perfect, computational) zero-knowledge proof systems against the honest verifier.

Remarks on the definitions.

1. (Honest verifiers) We only require that the zero-knowledge condition to hold against the honest verifier, *i.e.* the verifier that follows the protocol as specified. The usual definition requires the zero-knowledge property to hold against any polynomial-time verifier strategy. However, subsequent to this work, it has been shown that any proof system which is statistical zero knowledge against the honest verifier can be transformed into one that is zero knowledge against cheating verifiers [GSV98]. Via this transformation, many of our results directly translate to the class of promise problems possessing statistical zero-knowledge proofs against cheating verifiers. This is discussed in detail in Section 5.

2. (Error probabilities) The completeness and soundness error probabilities can be made exponentially small without increasing the number of rounds, because zero-knowledge *against an honest verifier* is preserved under parallel repetition.

3. (Strict polynomial-time simulation) Following [Gol95], we work with the variant of zeroknowledge in which the simulator is required to run in *strict* polynomial time, with some probability of failure in the perfect case. The original definition in [GMR89] allows the simulator to run in expected polynomial time, but with zero probability of failure. Our choice is not very restrictive, because we are only discussing honest-verifier statistical zero-knowledge and we do not know of any problems which require an expected polynomial time simulator for the honest verifier. In addition, as shown in Section 4.5, our techniques can be used to prove that expected polynomial time simulators and strict polynomial time simulators are actually *equivalent* for public-coin statistical zero-knowledge proofs against an honest verifier.

4. (Promise problems vs. languages) Our definitions above generalize the original definitions of [GMR89] from languages to promise problems, and we focus on the "promise class" SZK rather than the class of languages possessing statistical zero-knowledge proofs. A couple of justifications can be given for this extension. First, for essentially all of our results, the fact

[2] A failure probability can also be allowed in the definition of statistical zero-knowledge, but this can easily be reduced to an 2^{-n^k} for any constant k by repeated trials and absorbed in to the simulator deviation.

that we prove them for the promise class only makes them stronger, by virtue of the fact that the promise class contains the language class. Second, several of the most important natural problems known to be in SZK, such as those in [GK93, GG98], are not languages, but promise problems, so it may actually be preferable to study the promise class.

Our only result which requires new interpretation for the language class is the Completeness Theorem. As the complete problem is a promise problem, it is not complete for the language class in the usual sense. Nevertheless, it still gives a characterization of the language class, in that a language has a statistical zero-knowledge proof *if and only if* it reduces to the complete problem.

We note that one must be a bit more careful in a complexity-theoretic investigation of promise classes, particularly when discussing reductions that may violate the promise (cf., discussions in [ESY84, GG98]), and it may be the case that the language class has some different properties than the promise one.

3 The Completeness Theorem

3.1 The complete problem

The main aim of this paper is to demonstrate that SZK consists exactly of the problems that involve deciding whether two efficiently samplable distributions are either far apart or close together. This can be formally captured by the following promise problem Statistical Difference (abbreviated SD):

$$\begin{aligned} \mathrm{SD}_Y &= \left\{(C_0, C_1) : \|C_0 - C_1\| > \frac{2}{3}\right\} \\ \mathrm{SD}_N &= \left\{(C_0, C_1) : \|C_0 - C_1\| < \frac{1}{3}\right\} \end{aligned}$$

In the above definition, C_0 and C_1 are circuits; these define probability distributions as discussed in Section 2. The thresholds of $1/3$ and $2/3$ in this definition are not completely arbitrary; it is important for the Polarization Lemma of Section 3.2 that $(2/3)^2 > 1/3$.

We can now state the main theorem of the paper.

Theorem 3.1 (Completeness Theorem) Statistical Difference *is complete for* SZK.

The most striking thing about Theorem 3.1 is that it characterizes statistical zeroknowledge *with no reference to interaction or zero knowledge.* Future investigation of the class SZK can focus on the single problem SD, instead of dealing with arbitrarily complicated protocols, problems, and simulators.

We emphasize that the novelty of this result lies in the specific complete problem we present and not merely the *existence* of a complete promise problem. It is fairly straightforward to construct a complete promise problem for PZK involving descriptions of Turing machines for the verifier and simulator. (See Appendix B.) However, in contrast to SD, a complete problem constructed in this manner is essentially restatement of the definition of the class and therefore does not simplify the study of the class at all.

The proof of Theorem 3.1 will come in Sections 3.3 and 3.4 via two lemmas and a theorem of Okamoto [Oka00]. But first, we observe that a statement analogous to Theorem 3.1 can be made for BPP, if we generalize BPP to promise problems in the obvious way.

Proposition 3.2 *If* SD′ *is the promise problem obtained by modifying the definition of* SD *so that* C_0 *and* C_1 *only have 1 bit of output, then* SD′ *is complete for* BPP.

Proof: To see that SD′ is in BPP, first observe that for circuits C_0 and C_1 (or any random variables) that just output 0 or 1,

$$\|C_0 - C_1\| = |\Pr[C_0 = 1] - \Pr[C_1 = 1]|.$$

Thus, an estimate on $\|C_0 - C_1\|$ that is correct within an additive factor of 1/3 can be obtained by sampling C_0 and C_1 polynomially many times and counting the number of ones that occur for each. This is sufficient to decide SD′.

Now we show that every promise problem Π in BPP reduces to SD′. Let A be the PPT machine which outputs 1 with probability greater than 2/3 when $x \in \Pi_Y$, but outputs 1 with probability less than 1/3 when $x \in \Pi_N$. Let $p(n)$ be a polynomial bound on the running time of A. Given an input x, we can, by standard techniques,[3] produce in polynomial time a circuit C_x describing the computation of A on x for $p(|x|)$ steps. The input to C_x is the first $p(|x|)$ bits on the random tape of A the output is the first bit on the output tape. Let D be a circuit that always outputs 0. Then $\|C_x - D\| = \Pr[A(x) = 1]$, so $x \mapsto (C_x, D)$ is a polynomial-time reduction from Π to SD′. ■

Proposition 3.2 remains true even if we allow C_0 and C_1 to output strings of logarithmic length. Other classes such as P and co-RP can be obtained by modifying the definition of SD in a similar fashion (and changing the thresholds). This demonstrates that SZK is a natural generalization of these well-known classes.

3.2 A polarization lemma

In this section, we exhibit a transformation which "polarizes" the statistical relationship between two distributions. That is, pairs of distributions which are statistically close become much closer and pairs of distributions which are statistically far apart become much further apart.

Lemma 3.3 (Polarization Lemma)[4] *There is a polynomial-time computable function that takes a triple* $(C_0, C_1, 1^k)$*, where* C_0 *and* C_1 *are circuits, and outputs a pair of circuits* (D_0, D_1) *such that*

$$\|C_0 - C_1\| < 1/3 \quad \Rightarrow \quad \|D_0 - D_1\| < 2^{-k}$$
$$\|C_0 - C_1\| > 2/3 \quad \Rightarrow \quad \|D_0 - D_1\| > 1 - 2^{-k}$$

The usefulness of the Polarization Lemma comes from the fact that the two distributions it produces can be treated almost as if they were identically distributed or disjoint (*i.e.* statistical difference 0 and 1, respectively). Indeed, it will be essential in proving that SD (with thresholds of 2/3 and 1/3, as we've defined it) is in SZK and we will make further use of it in deriving consequences of Theorem 3.1.

Superficially, it may seem that a Chernoff bound argument is all that is needed to prove Lemma 3.3. However, Chernoff bounds are primarily useful for distinguishing between two events. This corresponds to *increasing* statistical difference, as formalized in the following "direct product" lemma:

[3] See, for example, [Pap94, Thms. 8.1 and 8.2].

[4] The Polarization Lemma stated here is called the Amplification Lemma in [SV97]. We change the name here to stress that the Polarization Lemma does not merely increase statistical difference.

Lemma 3.4 (Direct Product Lemma) *Let X and Y be distributions such that $\|X-Y\|=\epsilon$. Then for all k,*

$$k\epsilon \geq \| \otimes^k X - \otimes^k Y\| \geq 1 - 2e^{-k\epsilon^2/2}$$

Proof: The upper bound of $k\epsilon$ follows immediately from Fact 2.3, so we proceed to the proof of the lower bound. Recall, from the definition of statistical difference, that there must exist a set S such that

$$\Pr[X\in S] - \Pr[Y \in S] = \epsilon.$$

Let $p = \Pr[Y\in S]$, so $\Pr[X\in S] = p+\epsilon$. Hence, in k independent samples of X, the expected number of samples that lie in S is $(p+\epsilon)k$, whereas in k independent samples of Y, the expected number of samples that lie in S is pk. The Chernoff bound[5] tells us that the probability that *at least* $(p+\frac{\epsilon}{2})k$ components of $\otimes^k Y$ lie in S is at most $\exp(-k\epsilon^2/2)$, whereas the probability that *at most* $(p+\frac{\epsilon}{2})k$ components of $\otimes^k X$ lie in S is at most $\exp(-k\epsilon^2/2)$. Let S' be the set of all k-tuples that contain more than $(p+\frac{\epsilon}{2})k$ components that lie in S. Then,

$$\| \otimes^k X - \otimes^k Y\| \geq \Pr\left[\otimes^k X \in S'\right] - \Pr\left[\otimes^k Y \in S'\right] \geq 1 - 2e^{-k\epsilon^2/2}.$$

■

Note the gap between the upper and lower bounds in Lemma 3.4; the lower bound says that taking $O(1/\epsilon^2)$ copies is sufficient to increase statistical difference from ϵ to a constant, while the upper bound says that $\Omega(1/\epsilon)$ copies are necessary. This gap is inherent, and essentially amounts to the difference between 1-sided and 2-sided error: Taking X and Y to be distributions on $\{0,1\}$ that are 1 with probability 1 and $1-\epsilon$, respectively, we see that the statistical difference between $\otimes^k X$ and $\otimes^k Y$ is exactly $1-(1-\epsilon)^k$, which is a constant for $k=\Theta(1/\epsilon)$. On the other hand, when X and Y are 1 with probability $(1+\epsilon)/2$ and $(1-\epsilon)/2$, respectively, it can be shown that $k=\Theta(1/\epsilon^2)$ copies are necessary to increase the statistical difference to a constant. Furthermore, in this latter example, $\|X\otimes X - Y\otimes Y\| = \epsilon = \|X-Y\|$, so we cannot even hope to show that statistical difference always increases for every $k>1$ (as pointed out to us by Madhu Sudan).

Notice that the Direct Product Lemma 3.4 is *not* sufficient to prove the Polarization Lemma, because it always increases statistical difference, whereas we would like to increase statistical difference in some cases and decrease it in others. However, it does drive larger values of the statistical difference to 1 more quickly than it drives smaller values to 1, so it is a step in the right direction. The following lemma provides a complementary technique which decreases the statistical difference to 0, with small values going to 0 faster than large values.

Lemma 3.5 (XOR Lemma) *There is a polynomial-time computable function that maps a triple $(C_0, C_1, 1^k)$, where C_0 and C_1 are circuits, to a pair of circuits (D_0, D_1) such that $\|D_0 - D_1\| = \|C_0 - C_1\|^k$. Specifically, D_0 and D_1 are defined as follows:*

D_0: *Uniformly select $(b_1,\ldots,b_k)\in\{0,1\}^k$ such that $b_1\oplus\cdots\oplus b_k = 0$, and output a sample of $C_{b_1}\otimes\cdots\otimes C_{b_k}$.*
D_1: *Uniformly select $(b_1,\ldots,b_k)\in\{0,1\}^k$ such that $b_1\oplus\cdots\oplus b_k = 1$, and output a sample of $C_{b_1}\otimes\cdots\otimes C_{b_k}$.*

[5]For the formulation of the Chernoff bound we use, see, for example, the formulation of Hoeffding's inequality in [Hof95, Sec. 7.2.1].

In order to prove this lemma, we employ a generalization of the technique used in [DDPY94] to represent the logical AND of statements about Graph Nonisomorphism. This tool is described in the following Proposition.

Proposition 3.6 *Let X_0, X_1, Y_0, Y_1 be any random variables, and define the following pair of random variables:*

Z_0: *Choose $a, b \in_R \{0,1\}$ such that $a \oplus b = 0$. Output a sample of $X_a \otimes Y_b$.*
Z_1: *Choose $a, b \in_R \{0,1\}$ such that $a \oplus b = 1$. Output a sample of $X_a \otimes Y_b$.*

Then $\|Z_0 - Z_1\| = \|X_0 - X_1\| \cdot \|Y_0 - Y_1\|$.

The statistical difference between X_0 and X_1 (or Y_0 and Y_1) measures the advantage a computationally unbounded party has, over random guessing, in guessing b given a sample from X_b, where b is selected uniformly from $\{0,1\}$. (This view of statistical difference will become more apparent in the subsequent section.) Intuitively, the above Proposition says that the advantage one has in guessing the XOR of two independent bits is the product of the advantages one has for guessing each individual bit.

Proof:

$$\begin{aligned}
\|Z_0 - Z_1\| &= \frac{1}{2}\left|Z_0 - Z_1\right|_1 \\
&= \frac{1}{2}\left|\left(\frac{1}{2}X_0 \otimes Y_0 + \frac{1}{2}X_1 \otimes Y_1\right) - \left(\frac{1}{2}X_1 \otimes Y_0 + \frac{1}{2}X_0 \otimes Y_1\right)\right|_1 \\
&= \frac{1}{4}\left|(X_0 - X_1) \otimes (Y_0 - Y_1)\right|_1 \\
&= \left(\frac{1}{2}\left|X_0 - X_1\right|_1\right) \cdot \left(\frac{1}{2}\left|Y_0 - Y_1\right|_1\right) \\
&= \|X_0 - X_1\| \cdot \|Y_0 - Y_1\|
\end{aligned}$$

■

Proposition 3.6 and an induction argument establish Lemma 3.5. Yao's XOR Lemma [Yao82] (cf., [GNW95]) can be seen as an analogue of Lemma 3.5 in the computational setting, where the analysis is much more difficult.[6]

Now we combine the Direct Product and XOR constructions of Lemmas 3.4 and 3.5 to prove Lemma 3.3. The Direct Product Lemma gives a way to increase statistical difference with large values going to 1 faster than small values. Similarly, the XOR Lemma shows how to decrease statistical difference with small values going to 0 faster than large values. Intuitively, alternating these procedures should "polarize" large and small values of statistical difference, pushing them closer to 1 and 0, respectively. A similar alternation between procedures with complementary effects was used by Ajtai and Ben-Or [AB84] to amplify the success probability of randomized constant-depth circuits.

[6] To see the analogy, recall that Yao's XOR Lemma considers the maximum advantage an *efficient* algorithm has, over random guessing, in computing a bit b from string x when they are selected according to some distrbution $(b,x) \leftarrow (B,X)$ (e.g., X is uniform and B is a hardcore bit of $f^{-1}(X)$ for some one-way permutation f.). It states that the maximum advantage an efficient algorithm has in computing the XOR $b_1 \oplus \cdots \oplus b_k$ from $(x_1, \ldots, x_k)$ decreases exponentially with k when the pairs (b_i, x_i) are independentally distributed according to (B,X).

Proof: Let $\ell = \lceil \log_{4/3} 6k \rceil$. Apply Lemma 3.5 to the triple $(C_0, C_1, 1^\ell)$ to produce (C_0', C_1') such that if

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \left\|C_0' - C_1'\right\| < (1/3)^\ell \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \left\|C_0' - C_1'\right\| > (2/3)^\ell. \end{aligned}$$

Let $m = 3^{\ell-1}$. Let $C_0'' = \otimes^m C_0'$ and let $C_1'' = \otimes^m C_1'$. Then, by Fact 2.3 and the Direct Product Lemma,

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \left\|C_0'' - C_1''\right\| < 1/3 \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \left\|C_0'' - C_1''\right\| > 1 - 2\exp(-3^{\ell-1}(2/3)^{2\ell}/2) > 1 - 2e^{-k}. \end{aligned}$$

Finally, apply the transformation of Lemma 3.5 one more time to $(C_0'', C_1'', 1^k)$ to produce (D_0, D_1) such that

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \|D_0 - D_1\| < 3^{-k} < 2^{-k} \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \|D_0 - D_1\| > (1 - 2e^{-k})^k > 1 - 2ke^{-k} > 1 - 2^{-k}. \end{aligned}$$

■

Notice that the above analysis relies on the fact that $(2/3)^2 > (1/3)$, so it will not work if 2/3 and 1/3 are replaced by, say, .51 and .49. We do not know how to prove such a Polarization Lemma for arbitrary constant thresholds. We can however extend it to thresholds α and β, where $\alpha^2 > \beta$, and the running time will be polynomial in $\exp\left(\left(1 - \log(\alpha^2)/\log(\beta)\right)^{-1}\right)$ along with the input size. See [SV99] for more details.

3.3 A protocol for Statistical Difference

In this section, we show that SD has a simple two-message statistical zero-knowledge proof system, which is a generalization of the standard protocols for for Quadratic Nonresiduosity [GMR89] and Graph Nonisomorphism [GMW91]. Intuitively, if two distributions are statistically far apart, then, when given a random sample from one of the distributions, a computationally unbounded party should have a good chance of guessing from which distribution it came. However, if the two distributions are statistically very close, even a computationally unbounded party should not have much better than a 50% chance of guessing correctly. This suggests the following two-message (private-coin) protocol for SD:

Zero-knowledge Proof System for SD

Input: (C_0, C_1) (such that either $\|C_0 - C_1\| > 2/3$ or $\|C_1 - C_1\| < 1/3$)

1. V, P: Compute $(D_0, D_1) = \mathbf{Polarize}(C_0, C_1, 1^n)$, where $n = |(C_0, C_1)|$.
2. V: Flip one random coin $r \in \{0, 1\}$. Let z be a sample of D_r. Send z to P.
3. P: If $\Pr[D_0 = z] > \Pr[D_1 = z]$, answer 0, otherwise answer 1.
4. V: Accept if P's answer equals r, reject otherwise.

Lemma 3.7 *The above is a statistical zero-knowledge proof system for* SD*, with soundness error* $\frac{1}{2} + 2^{-n}$*, and completeness error and simulator deviation both* 2^{-n}*. Thus* $\mathrm{SD} \in \mathsf{SZK}$.

Proof: We will argue that the prover strategy given in the protocol is optimal (*i.e.* maximizes the verifier's acceptance probability), and use this to bound both the soundness and completeness error. The simulator deviation will then follow easily.

Consider any prover P^*. Suppose for some z the prover P^* fails to follow the strategy we present. If $\Pr[D_0 = z] \neq \Pr[D_1 = z]$, this means that with nonzero probability, P^* choses the distribution in which z is less likely to occur. Then, conditioned on z, the success probability of P^* will certainly be lower than that of the prover in our protocol. If $\Pr[D_0 = z] = \Pr[D_1 = z]$, the prover has no information about r, so no matter what strategy it uses, it has exactly even odds of guessing correctly. Since these observations hold for all z, the given prover is optimal.

We now analyze the probability of success of the optimal prover. Recall that $\|D_0 - D_1\| = \Pr[D_0 \in S] - \Pr[D_1 \in S]$ for $S = \{z : \Pr[D_0 = z] > \Pr[D_1 = z]\}$. The probability that the optimal prover guesses correctly is exactly

$$\begin{aligned} \frac{1}{2}\Pr\left[D_0 \in S\right] + \frac{1}{2}\Pr\left[D_1 \notin S\right] &= \frac{1}{2}\left(\Pr\left[D_0 \in S\right] + 1 - \Pr\left[D_1 \in S\right]\right) \\ &= \frac{1 + \|D_0 - D_1\|}{2}. \end{aligned}$$

By Lemma 3.3, $\|D_0 - D_1\| > 1 - 2^{-n}$ when (C_0, C_1) is a YES instance of SD, and $\|D_0 - D_1\| < 2^{-n}$ when (C_0, C_1) is a NO instance. Hence, the probability that the prover convinces the verifier to accept is greater than $(1 + 1 - 2^{-n})/2 > 1 - 2^{-n}$ for YES instances, and less than $(1 + 2^{-n})/2 < 1/2 + 2^{-n}$ for NO instances. This immediately gives the completeness error; the soundness error also follows because we considered the optimal prover strategy.

Now, notice that when the prover answers correctly, all the verifier receives from the prover is the value of r, which the verifier already knew. Thus, since we have shown that the prover is answering correcty with all but exponentially small probability, intuitively the verifier learns nothing. To turn this intuition into a proof of statistical zero knowledge, we consider the following probabilistic polynomial-time simulator: On input (C_0, C_1), the simulator first computes $(D_0, D_1) = \textbf{Polarize}(C_0, C_1, 1^n)$, where $n = |(C_0, C_1)|$. The simulator then flips one random coin $r \in \{0, 1\}$. If $r = 0$, it samples z from D_0, otherwise it samples z from D_1. The simulator then outputs a conversation in which the verifier sends z to the prover, and the prover responds with r. The simulator also outputs the random coins it used to generate r and z as the coins of the verifier. Thus, the simulator presented here always outputs conversations in which the prover responds correctly. Except for the prover's response, all other components of the simulator's output distribution are distributed identically to the verifier's view of the real interaction. Hence, the simulator deviation is bounded by the probability that the prover responds incorrectly in the real interaction, which we have already argued is at most 2^{-n} in the case of YES instances. ■

Note that the above proof system remains complete and sound even without polarization, but for the zero-knowledge property, we need to make the statistical difference very close to 1 on YES instances.

By using a security parameter k rather than n in the call to **Polarize**, both the completeness error and simulator deviation can be reduced to 2^{-k}. Thus, even very short assertions about SD can be proven with with very high security. Contrast this with the original definition of SZK [GMR89], which only requires that the simulator deviation vanish as an *negligible* function of the *input length.* This property has obvious cryptographic significance, so we formulate it more precisely in Section 4.1.

3.4 SZK-hardness of SD

The other main lemma we prove to show that SD is complete for SZK follows:

Lemma 3.8 *Suppose promise problem Π has a public coin statistical zero-knowledge proof system. Then there exist PPT's A and B and a negligible function α such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow \|A(x) - B(x)\| \leq \alpha(|x|), \text{ and} \\ x \in \Pi_N &\Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}. \end{aligned}$$

We defer the proof of this lemma to Section 3.5, and first observe how it gives a reduction to SD for problems with public-coin statistical zero-knowledge proofs.

Corollary 3.9 *Suppose promise problem Π has a public-coin statistical zero-knowledge proof system. Then Π reduces to $\overline{\text{SD}}$. (Equivalently, $\overline{\Pi}$ is reduces to* SD*.)*

Proof: First apply Lemma 3.8 to obtain A and B, with $p(|x|)$ being a polynomial bound on the running times of $A(x)$ and $B(x)$. Given a string x, we can, by standard techniques,[7] produce in polynomial time circuits C_0 and C_1 describing the computation of A and B, respectively, on x for $p(|x|)$ steps. The inputs to C_0 and C_1 are the first $p(|x|)$ bits on the random tapes of A and B and the outputs are the first $p(|x|)$ positions on the output tapes. Then $\|C_0 - C_1\| = \|A(x) - B(x)\|$, which is at most $\alpha(|x|) < 1/3$ if $x \in \Pi_Y$ and at least $1 - 2^{-|x|} > 2/3$ if $x \in \Pi_N$ (for all sufficiently long x). So $x \mapsto (C_0, C_1)$ is a reduction from Π to $\overline{\text{SD}}$ (for all but finitely many x). ■

The final ingredient in the proof of Theorem 3.1 is a theorem of Okamoto [Oka00], which we state in terms of promise problems.[8]

Theorem 3.10 ([Oka00, Thm. 1]) *If a promise problem Π has a statistical zero-knowledge proof system, then Π has a* public-coin *statistical zero-knowledge proof system.*

Now it will be easy to show that SD is complete for SZK.

Proof of Theorem 3.1: Lemma 3.7 tells us that SD $\in$ SZK, so we only need to show that every problem in SZK reduces to SD. Corollary 3.9 and Theorem 3.10 imply that every problem $\Pi \in$ SZK reduces to $\overline{\text{SD}}$. In particular, SD reduces to $\overline{\text{SD}}$, or, equivalently, $\overline{\text{SD}}$ reduces to SD. Composing reductions, it follows that every problem $\Pi \in$ SZK reduces to SD. ■

3.5 Proof of Lemma 3.8

The constructions in this lemma and the statistical zero-knowledge proof system for STATISTICAL DIFFERENCE are carried out for the specific example of GRAPH ISOMORPHISM in Appendix C.

[7] See, for example, [Pap94, Thms. 8.1 and 8.2].

[8] Okamoto stated his result in terms of languages, but the proof readily extends to promise problems (cf., [GV99]).

Intuition. Recall that we wish to construct a pair of probabilistic polynomial-time machines A and B such that if $x \in \Pi_Y$, the distributions $A(x)$ and $B(x)$ are statistically very close, but when $x \in \Pi_N$, $A(x)$ and $B(x)$ are far apart. We are given that Π has a *public-coin* statistical zero-knowledge proof system. A natural place to search for the desired distributions is in the output of the simulator for this proof system. We think of the simulator as describing the moves of a *virtual prover* and a *virtual verifier.*[9]. We wish to find properties of the simulator's output that (1) distinguish the case $x \in \Pi_Y$ from $x \in \Pi_N$, and (2) are captured by the statistical difference between samplable distributions. In the case that $x \in \Pi_Y$, we have strong guarantees on the simulator's output. Namely, it outputs accepting conversations with high probability and its output distribution is statistically very close to the real interaction. When $x \in \Pi_N$, there are two cases. If the simulator outputs accepting conversations with low probability, this easily distinguishes it from the simulator output when $x \in \Pi_Y$. However, it is possible that the simulator will output accepting conversations with high probability even when $x \in \Pi_N$. This means that the virtual prover is doing quite well in fooling the virtual verifier. This naturally suggest a strategy for a real prover — imitate the virtual prover's behavior. Such a prover, called a *simulation-based prover*, was introduced by Fortnow [For89] and is a crucial construct in our proof. The soundness of the proof system tells us that the simulation-based prover cannot hope to convince the real verifier with high probability. There must be a reason for this discrepancy between the success rates of the virtual prover and the simulation-based prover. One possibility is that the virtual verifier's coins in the simulator's output are *far from uniform*, so that the simulation only captures a small fraction of possible verifier states. However, this is not the only difficulty. A second difficulty is that the responses of the virtual prover may depend on future coin tosses of the virtual verifier, which is impossible in a real public-coin interaction. Note that this is equivalent to the virtual verifier's coin tosses being *dependent on previous messages* of the virtual prover. We will show that these are the only two obstacles the simulation-based prover faces in trying to fool the verifier, and thus they must be present when $x \in \Pi_N$. In the case that $x \in \Pi_Y$, however, these difficulties cannot arise since we are guaranteed that the simulator output distribution is very close to that of the real interation. If we could measure the extent to which these anomalies are present by the statistical difference between samplable distributions, we would achieve our objective. This is precisely what we do.

Notation. Let (P, V) be a public-coin interactive proof system for a promise problem Π which is (honest-verifier) statistical zero knowledge and let S be a simulator for this proof system. Without loss of generality, we may assume that the interaction of P and V on input x always has $2r(|x|)$ exchanged messages, with V sending the first message and each message consisting of exactly $q(|x|)$ bits, for some polynomials q and r. Moreover, it may be assumed that S's output always consists of $2r(|x|)$ strings of length $q(|x|)$. The output of S and the conversation between P and V on input x will be written in the form $S(x) = (c_1, p_1, \ldots, c_r, p_r)_S$ and $(P, V)(x) = (c_1, p_1, \ldots, c_r, p_r)_{(P,V)}$, respectively, where $c_1, \ldots, c_r$ represent the messages (equivalently coin tosses, since we are in the public-coin setting) of V, $p_1, \ldots, p_r$ represent the prover messages, and $r = r(|x|)$. (Dependence on x will often be omitted in this manner for notational convenience.) We use notation such as $(c_i)_S$ for the random variable obtained by running S once and taking the c_i-component of its output. More generally, partial conversation transcripts will be written like $(c_1, p_1, c_2, p_2)_S$. We call a conversation transcript $(c_1, p_1, \ldots, c_r, p_r)$ which would make V accept (resp., reject) an *accepting conversation*

[9] This terminology is taken from [AH91]. The cases we consider are quite similar to those analyzed in [For89, AH91] Because we focus on public-coin proofs, many complications faced in those works do not arise. This allows us to make some new observations and reach a novel conclusion (namely, the Completeness Theorem).

(resp., *rejecting conversation*). We denote by $U(n)$ the uniform distribution on strings of length n.

The proof. In order to formalize the above intuition, a definition of the simulation-based prover needs to be given. This is the prover P^* that imitates the virtual prover, *i.e.* P^* does the following to compute its next message when the current conversation transcript is $(c_1, p_1, \ldots, c_i)$:

If $S(x)$ outputs conversations that begin with $(c_1, p_1, \ldots, c_i)$ with probability 0, then output $0^{q(|x|)}$.

Else output $y \in \{0,1\}^{q(|x|)}$ with probability

$$p_y = \Pr[S(x) \text{ begins with } (c_1, p_1, \ldots, c_i, y) | S(x) \text{ begins with } (c_1, p_1, \ldots, c_i)].$$

In order to analyze the success probability of P^*, we first compare the output of S to the actual conversations between P^* and V. Let ϵ_i be the statistical difference between $(c_1, p_1 \ldots, c_{i-1}, p_{i-1}, c_i)_S$ and $(c_1, p_1 \ldots, c_{i-1}, p_{i-1})_S \otimes U(q(|x|))$. Thus ϵ_i measures how far from uniform the virtual verifier's i-th set of coins are and how far from independent they are from what comes before. The following claim formalizes our intuition that P^* can do as well as the virtual prover, as long as the virtual verifier's coins are near-uniform and near-independent from what preceeds them.

Claim 3.11 $\|S(x) - (P^*, V)(x)\| \leq \sum_{i=0}^{r} \epsilon_i$.

Proof of claim: Let $C_i^S = (c_1, p_1, \ldots, c_i)_S$ be the random variable of partial simulator transcripts ending with the i-th coins of the virtual verifier. Let $P_i^S = (c_1, p_1, \ldots, c_i, p_i)_S$ be the random variable of partial transcripts ending with the i-th virtual prover response. Similarly define C_i^* and P_i^* as partial conversation transcripts of (P^*, V). The aim is to show that at round k, the statistical difference grows by at most ϵ_k. Formally, it will be shown by induction on k that

$$\left\|P_k^S - P_k^*\right\| \leq \sum_{i=0}^{k} \epsilon_i$$

The case $k = 0$ is trivial. For general k, first note that since P^* gives a response chosen according to the same distribution as the virtual prover, adding these responses to the conversations cannot increase the statistical difference (by Fact 2.4). That is,

$$\left\|P_{k+1}^S - P_{k+1}^*\right\| = \left\|C_{k+1}^S - C_{k+1}^*\right\|.$$

The idea now is to extract the parts of $\|C_{k+1}^S - C_{k+1}^*\|$ corresponding to ϵ_{k+1} and observe that what is left is simply the error from the previous round. Note that $C_{k+1}^* = P_k^* \otimes U(q(|x|))$, since the real verifier's coins are always uniform and independent from what came before.

Then, applying Fact 2.3 and the Triangle Inequality,

$$\begin{aligned}
\|C_{k+1}^S - C_{k+1}^*\| &\leq \left\|C_{k+1}^S - P_k^S \otimes U(q(|x|))\right\| + \left\|P_k^S \otimes U(q(|x|)) - P_k^* \otimes U(q(|x|))\right\| \\
&\leq \epsilon_{k+1} + \left\|P_k^S - P_k^*\right\| + \|U(q(|x|)) - U(q(|x|))\| \\
&\leq \epsilon_{k+1} + \sum_{i=0}^{k} \epsilon_i.
\end{aligned}$$

This completes the induction. Since $P_r^S = S(x)$ and $P_r^* = (P^*, V)(x)$, the Claim is proved.

Algorithm A		Algorithm B	
$A_0(x)$	Run $S(x)$ for $\lvert x\rvert$ repetitions. Output '1' if the majority are accepting conversations and '0' otherwise.	$B_0(x)$	Output 1.
$A_i(x)$	Run $S(x)$ to output $(c_1, p_1, \ldots, c_i)_{S(x)}$.	$B_i(x)$	Run $S(x)$ and flip $q(\lvert x\rvert)$ more coins to output $(c_1, p_1, \ldots, c_{i-1}, p_{i-1})_{S(x)} \otimes U(q(\lvert x\rvert))$.

Table 1: The components of A and B

We are now ready to construct the distributions we seek. The two distributions A and B each consist of $r + 1$ components, shown in Table 1. A is the algorithm whose output on input x is $(A_0(x), A_1(x), \ldots, A_r(x))$, all run independently, and B is the algorithm whose output is $(B_0(x), B_1(x), \ldots, B_r(x))$, all run independently.

Here, A_i is a sampling of a partial conversation transcript from S up to the virtual verifier's i-th set of coins, while B_i is a sampling of a partial conversation transcript from S up to the virtual prover's $(i-1)$-st response followed by $q(|x|)$ independent random bits. So, for $i \geq 1$, the statistical difference between A_i and B_i is ϵ_i.

We will show that the statistical difference between A and B is negligible if $x \in \Pi_Y$ and is noticeable if $x \in \Pi_N$. Amplifying this gap by repetition will yield Lemma 3.8.

Claim 3.12 *There exists a negligible function α such that if $x \in \Pi_Y$, then $\|A(x) - B(x)\| \leq \alpha(|x|)$.*

Proof of claim: The statistical difference between $A(x)$ and $B(x)$ is bounded above by the sum of the statistical differences between $A_i(x)$ and $B_i(x)$ over $i = 1, \ldots, r(|x|)$ (by Fact 2.3). First, let's examine $i = 0$. Since $S(x)$ outputs a conversation which makes V accept with probability at least $2/3 - \mathrm{neg}(|x|)$, the Chernoff bound implies that $\Pr[A_0(x) = 1] = 1 - 2^{-\Omega(|x|)}$, so the statistical difference between A_0 and B_0 is negligible. For $i \geq 1$, recall that in the real conversations of P and V, the verifier's coins are truly uniform and independent from prior rounds, so $\|A_i(x) - B_i(x)\|$ should essentially be bounded by the statistical difference between the simulator's output and the real interaction. This is in fact true, as (omitting x from the notation):

$$\begin{aligned} \|A_i - B_i\| &\leq \|A_i - (c_1, p_1, \ldots, c_i)_{P,V}\| + \|(c_1, p_1, \ldots, c_i)_{P,V} - B_i\| \\ &\leq \|S - (P, V)\| + \|S - (P, V)\|. \end{aligned}$$

(The last inequality is by Fact 2.4.) Thus,

$$\|A(x) - B(x)\| \leq 2^{-\Omega(|x|)} + 2r(|x|) \cdot \|S(x) - (P, V)(x)\|,$$

which is negligible since $\|S(x) - (P, V)(x)\|$ is negligible and $r(x)$ is polynomial.

Claim 3.13 *If $x \in \Pi_N$ then $\|A(x) - B(x)\| \geq 1/12r(|x|)$.*

Proof of claim: It suffices to show that for some i, $\epsilon_i = \|A_i(x) - B_i(x)\| > 1/12r(|x|)$ (by Fact 2.4). We deal with two cases depending on the probability that S outputs an accepting conversation.

Case 1: $\Pr[S(x) \text{ accepts }] \leq 5/12$. Then, by the Chernoff bound, $\Pr[A_0(x) = 1] \leq$

$2^{-\Omega(|x|)}$, so the statistical difference between $A_0(x)$ and $B_0(x)$ is at least $1 - 2^{-\Omega(|x|)} > 1/12r(|x|)$.

Case 2: $\Pr[S(x) \text{ accepts }] > 5/12$. Then, since $\Pr[(P^*, V)(x) \text{ accepts }]$ is at most $1/3$, we must have

$$\sum_{i=0}^{r} \epsilon_i \geq \|S(x) - (P^*, V)(x)\| > \frac{5}{12} - \frac{1}{3} = \frac{1}{12}.$$

Thus, at least one ϵ_i must be greater than $1/12r(|x|)$.

Now consider the samplable distributions $\hat{A}(x) = \otimes^{s(|x|)} A(x)$ and $\hat{B}(x) = \otimes^{s(|x|)} B(x)$, where $s(n) = n \cdot r(n)^2$. If $x \in \Pi_Y$, $\|\hat{A}(x) - \hat{B}(x)\| \leq s(|x|) \cdot \|A(x) - B(x)\|$, which is still negligible. If $x \in \Pi_N$, then by the Direct Product Lemma (Lemma 3.4), $\|\hat{A}(x) - \hat{B}(x)\| \geq 1 - 2^{-\Omega(|x|)}$. This completes the proof of Lemma 3.8. ■

4 Applications

4.1 Efficient statistical zero-knowledge proofs

The proof system for Statistical Difference given in Section 3.3 has a number of desirable features. It is very efficient in terms of communication and interaction, and the simulator deviation can be made exponentially small in a security parameter (that can be varied independently of the input length). By the Completeness Theorem, it follows that every problem in SZK also has a proof system with these properties.

We begin by formalizing one of the properties of the SD proof system that was informally discussed in Section 3.3.

Definition 4.1 *An interactive protocol (P, V) is called a* security-parametrized statistical zero-knowledge proof system *for a promise problem Π if there exists a PPT simulator S, a negligible function $\alpha(k)$ (called the* simulator deviation*), and completeness and soundness errors $c(k)$ and $s(k)$ such that for all strings x and all $k \in \mathbb{N}$,*

1. *If $x \in \Pi_Y$, then $\Pr\left[(P, V)(x, 1^k) = \texttt{accept}\right] \geq 1 - c(k)$.*
2. *If $x \in \Pi_N$, then for all P^*, $\Pr\left[(P^*, V)(x, 1^k) = \texttt{accept}\right] \leq s(k)$.*
3. *If $x \in \Pi_Y$, then $\left\|S(x, 1^k) - \mathrm{View}_{P,V}(x, 1^k)\right\| \leq \alpha(k)$.*

As usual, we require that $c(k)$ and $s(k)$ are computable in time $\mathrm{poly}(k)$ and $1 - c(k) > s(k) + 1/\mathrm{poly}(k)$

We now describe the efficient proof systems inherited by all of SZK.

Corollary 4.2 *Every problem in SZK possesses a security-parametrized statistical zero-knowledge proof system with the following properties:*

1. *Simulator deviation 2^{-k}, completeness error 2^{-k}, and soundness error $1/2 + 2^{-k}$.*
2. *The prover and verifier exchange only 2 messages.*

3. The prover sends only 1 bit to the verifier.

4. The prover is deterministic.

Proof: Let Π be any promise problem in SZK. Let f be the reduction from Π to SD guaranteed by the Completeness Theorem. A protocol with the desired properties for Π can be obtained as follows: on input $(x, 1^k)$, execute the proof system for SD, given in Section 3.3, on input $f(x)$ and using k rather than n in the call to **Polarize**. ■

4.2 Closure properties

In this section, we prove several closure properties of SZK. The first, closure under reductions, is a direct consequence of the "security parametrization" property shown to hold for SZK in the previous section.

Corollary 4.3 SZK *is closed under (Karp) reductions. That is, if* $\Pi \in \mathsf{SZK}$ *and* Γ *reduces to* Π, *then* $\Gamma \in \mathsf{SZK}$.

Proof: By Corollary 4.2, Π has a security-parameterized statistical zero-knowledge proof. A statistical zero-knowledge proof for Γ can be obtained as follows: On input x, the prover, verifier, and simulator run the security-parametrized proof for Π on input $(f(x), 1^{|x|})$, where f is the reduction from Γ to Π. ■

The security-parametrization property is essential in the above proof, because an arbitrary reduction f could potentially shrink string lengths dramatically, and we want the simulator deviation to be negligible as a function of $|x|$, not $|f(x)|$.

Next, we show how Okamoto's result that SZK is closed under complement follows immediately from our proof of Completeness Theorem.

Corollary 4.4 ([Oka00, Thm. 2]) SZK *is closed under complement, even for promise problems.*

Proof: Let Π be any problem in SZK. By Theorem 3.10 and Corollary 3.9, $\overline{\Pi}$ reduces to SD, which is in SZK. By Corollary 4.3, $\overline{\Pi} \in \mathsf{SZK}$. ■

Before moving on to additional closure properties, we deduce the upper bounds of Fortnow [For89] and Aiello and Håstad [AH91] on the complexity of SZK.

Corollary 4.5 ([For89, AH91]) $\mathsf{SZK} \subset \mathsf{AM} \cap \mathsf{co\text{-}AM}$, *where* AM *denotes the class of problems possessing constant-message interactive proofs.*

Proof: Immediate from Corollaries 4.2 and 4.4. ■

Above, we have shown that SZK satisfies a computational closure property (Corollary 4.3) and a boolean closure property (Corollary 4.4). Now we will exhibit a stronger closure property, which can be viewed as both a computational one and a boolean one: given an arbitrary boolean formula whose atoms are statements about membership in *any* problem in SZK, one can efficiently construct a statistical zero-knowledge interactive proof for its validity. Note that such a property does not follow immediately from the fact that a class is closed under intersection, union, and complementation, because applying these more than a constant number of times could incur a superpolynomial cost in efficiency, while we ask that the construction can be done efficiently with respect to the size of the formula. The procedure for doing this is based on work by De Santis, Di Crescenzo, Persiano, and Yung [DDPY94]. They show how to construct statistical zero-knowledge proofs for all monotone boolean formulae whose atoms are statements about a random self-reducible language. Their zero-knowledge proofs are constructed by producing two distributions which are either disjoint or identical, depending on whether or not the formula is true. Hence, their construction can be viewed as a reduction to an extreme case of SD, in which the thresholds are 1 and 0.

Using the Direct Product, XOR, and the Polarization Lemmas of Section 3.2, we generalize their result to monotone formulae whose atoms are statements about membership in Statistical Difference. Then, using the completeness of SD (Theorem 3.1) and closure under complement (Corollary 4.4), we deduce the result for general (*i.e.* non-monotone) formulae and every promise problem in SZK.

We begin with some definitions describing precisely what kind of boolean closure properties we will achieve. (Later, we will see how it can also be interpreted as closure under a certain class of polynomial-time reductions.) In order to deal with instances of promise problems that violate the promise, we will work with an extension of boolean algebra that includes an additional "ambiguous" value $\star$.

Definition 4.6 *A* partial assignment *to variables $v_1, \ldots, v_k$ is a k-tuple $\overline{a} = (a_1, \ldots, a_k) \in \{0, 1, \star\}^k$. For a propositional formula (or circuit) ϕ on variables $v_1, \ldots, v_k$, the evaluation $\phi(\overline{a})$ is recursively defined as follows:*

$$v_i(\overline{a}) = a_i \qquad (\phi \wedge \psi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 1 \text{ and } \psi(\overline{a}) = 1 \\ 0 & \text{if } \phi(\overline{a}) = 0 \text{ or } \psi(\overline{a}) = 0 \\ \star & \text{otherwise} \end{cases}$$

$$(\neg\phi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 0 \\ 0 & \text{if } \phi(\overline{a}) = 1 \\ \star & \text{if } \phi(\overline{a}) = \star \end{cases} \qquad (\phi \vee \psi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 1 \text{ or } \psi(\overline{a}) = 1 \\ 0 & \text{if } \phi(\overline{a}) = 0 \text{ and } \psi(\overline{a}) = 0 \\ \star & \text{otherwise} \end{cases}$$

Note that $\phi(\overline{a})$ equals 1 (resp., 0) for some partial assignment $\overline{a}$, then $\phi(\overline{a}')$ also equals 1 (resp., 0) for every *boolean* $\overline{a}'$ obtained by replacing every $\star$ in $\overline{a}$ with either a 0 or 1. The converse, however, is not true: The formula $\phi = v \vee \neg v$ evaluates to 1 on every boolean assignment, yet is not 1 when evaluated at $\star$. Thus, the "law of excluded middle" $\phi \vee \neg\phi \equiv 1$ no longer holds in this setting. However, other identities in boolean algebra such as De Morgan's laws (e.g. $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$) do remain true.

Definition 4.7 *For a promise problem Π, the* characteristic function *of Π is the map $\chi_\Pi : \{0,1\}^* \to \{0, 1, \star\}$ given by*

$$\chi_\Pi(x) = \begin{cases} 1 & \text{if } x \in \Pi_Y \\ 0 & \text{if } x \in \Pi_N \\ \star & \text{otherwise} \end{cases}$$

Definition 4.8 *For any promise problem Π, we define a new promise problem $\Phi(\Pi)$ as follows: ϕ*

$$\begin{aligned} \Phi(\Pi)_Y &= \{(\phi, x_1, \ldots, x_k) : \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_k)) = 1\} \\ \Phi(\Pi)_N &= \{(\phi, x_1, \ldots, x_k) : \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_k)) = 0\}, \end{aligned}$$

where ϕ is a k-ary propositional formula. Mon(Π) *is defined analogously, except that only monotone ϕ are considered.*[10]

[10] In [DDPY94], only monotone formulae are treated. What they call $\Phi(L)$ is what we call Mon(L).

Sample(ψ, b)
If $\psi = v_i$, sample $z \leftarrow D^i_b$.
If $\psi = \tau \vee \mu$,
Sample $z_1 \leftarrow$ Sample(τ, b);
Sample $z_2 \leftarrow$ Sample(μ, b);
Let $z = (z_1, z_2)$.
If $\psi = \tau \wedge \mu$,
Choose $c, d \in_R \{0,1\}$ subject to $c \oplus d = b$;
Sample $z_1 \leftarrow$ Sample(τ, c);
Sample $z_2 \leftarrow$ Sample(μ, d);
Let $z = (z_1, z_2)$.
Output z.

Figure 1:

In [DDPY94], it is shown that $\mathrm{Mon}(L) \in \mathsf{SZK}$ for any language L which is random self-reducible, whose complement is self-reducible, or whose complement has a noninteractive statistical zero-knowledge proof. They also give statistical zero-knowledge proofs for some simple statements involving a random-self-reducible language and its complement. Damgård and Cramer [DC96] extend these results by showing that $\mathrm{Mon}(L) \in \mathsf{SZK}$ as long as L or its complement has a 3-message public-coin statistical zero-knowledge proof, and also treat a larger class of monotone functions.

Our result holds for all of SZK and for all boolean formulae, not just monotone ones:

Theorem 4.9 *For any promise problem* $\Pi \in \mathsf{SZK}$, $\Phi(\Pi) \in \mathsf{SZK}$.

This theorem can be generalized to work for all boolean formulae whose atoms are statements about membership in any finite set of languages in SZK, but we omit the notationally cumbersome formal statement since it is immediate from the completeness of Statistical Difference.

The main step in proving Theorem 4.9 is the following Lemma, which is based on the construction of [DDPY94] for Mon(Graph Nonisomorphism):

Lemma 4.10 $\mathrm{Mon}(\mathrm{SD}) \in \mathsf{SZK}$.

Proof: For intuition, consider two instances of statistical difference (C_0, C_1) and (D_0, D_1), both of which have statistical difference very close to 1 or very close to 0 (which can be achieved by the Polarization Lemma). Then $(C_0 \otimes D_0, C_1 \otimes D_1)$ will have statistical difference very close to 1 if either of the original statistical differences is very close to 1 and will have statistical difference very close to 0 otherwise. Thus, this Direct Product operation represents OR. Similarly, the XOR operation in Proposition 3.6 represents AND. We will recursively apply these constructions to obtain a reduction from Mon(SD) to SD. By closure under reductions (Corollary 4.3), Lemma 4.10 will follow.

Let $w = (\phi, (C^1_0, C^1_1), \ldots, (C^k_0, C^k_1))$ be an instance of Mon(SD) and let $n = |w|$. By applying the Polarization Lemma (Lemma 3.3), we can constuct in polynomial time pairs of circuits $(D^1_0, D^1_1), \ldots, (D^k_0, D^k_1)$ such that the statistical difference between D^i_0 and D^i_1 is greater than $1 - 2^{-n}$ if $(C^i_0, C^i_1) \in \mathrm{SD}_Y$ and is less than 2^{-n} if $(C^i_0, C^i_1) \in \mathrm{SD}_N$.

Consider the randomized recursive procedure Sample(ψ, b) in Figure 1 which takes a subformula ψ of $\phi = \phi(v_1, \ldots, v_n)$ and a bit $b \in \{0,1\}$ as input. Executing Sample(ϕ, b) for $b \in \{0,1\}$ takes time polynomial in n, because the number of recursive calls is equal to the number of subformulas of ϕ. For a subformula ψ of ϕ, define

$$\mathrm{Dif}(\psi) = \|\mathrm{Sample}(\psi, 0) - \mathrm{Sample}(\psi, 1)\|.$$

Then we can prove the following about Dif:

Claim 4.11 *Let $\overline{a} = (\chi_{\mathrm{SD}}(C_0^1, C_1^1), \ldots, \chi_{\mathrm{SD}}(C_0^k, C_1^k))$. For every subformula ψ of ϕ, we have:*

$$\begin{aligned} \psi(\overline{a}) = 1 \quad &\Rightarrow \quad \mathrm{Dif}(\psi) > 1 - |\psi| 2^{-n} \\ \psi(\overline{a}) = 0 \quad &\Rightarrow \quad \mathrm{Dif}(\psi) < |\psi| 2^{-n} \end{aligned}$$

Note that nothing is claimed when $\psi(\overline{a}) = \star$.

> **Proof of claim:** The proof of the claim is by induction on subformulae ψ of ϕ. It holds for atomic subformulae (*i.e.* the variables v_i) by the properties of the D_b^i's.
>
> **Case I:** $\psi = \tau \vee \mu$. If $\psi(\overline{a}) = 1$, then either $\tau(\overline{a}) = 1$ or $\mu(\overline{a}) = 1$. Without loss of generality, say $\tau(\overline{a}) = 1$. Then, by Fact 2.4 and induction,
>
> $$\mathrm{Dif}(\psi) \geq \mathrm{Dif}(\tau) > 1 - |\tau| 2^{-n} > 1 - |\psi| 2^{-n}.$$
>
> If $\psi(\overline{a}) = 0$, then $\tau(\overline{a}) = \mu(\overline{a}) = 0$. By Fact 2.3 and induction,
>
> $$\mathrm{Dif}(\psi) \leq \mathrm{Dif}(\tau) + \mathrm{Dif}(\mu) < |\tau| 2^{-n} + |\mu| 2^{-n} \leq |\psi| 2^{-n}.$$
>
> **Case I:** $\psi = \tau \wedge \mu$. By Proposition 3.6, $\mathrm{Dif}(\psi) = \mathrm{Dif}(\tau) \cdot \mathrm{Dif}(\mu)$. If $\psi(\overline{a}) = 1$, then, by induction,
>
> $$\mathrm{Dif}(\psi) \geq (1 - |\tau| 2^{-n})(1 - |\mu| 2^{-n}) > 1 - (|\tau| + |\mu|) 2^{-n} \geq 1 - |\psi| 2^{-n}.$$
>
> If $\psi(\overline{a}) = 0$, then, without loss of generality, say $\tau(\overline{a}) = 0$. By induction,
>
> $$\mathrm{Dif}(\psi) \leq \mathrm{Dif}(\tau) < |\tau| 2^{-n} \leq |\psi| 2^{-n}.$$

Now, let A and B be the circuits which sample from the distributions $\mathrm{Sample}(\phi, 0)$ and $\mathrm{Sample}(\phi, 1)$, respectively. (The the random bits each procedure uses are the inputs to the circuits). By the above claim, $\|A - B\| > 1 - n2^{-n} > 2/3$ if $\phi(\overline{a}) = 1$, and $\|A - B\| < n2^{-n} < 1/3$ if $\phi(\overline{a}) = 0$. In other words, the construction of A and B from w is a reduction from Mon(SD) to SD. This reduction can be computed in polynomial time because Sample runs in polynomial time. Thus, by Corollary 4.3, $\mathrm{Mon(SD)} \in \mathsf{SZK}$. ■

Now it is straightforward to deduce Theorem 4.9.

Proof: Let Π be any promise problem in SZK. By closure under complement (Corollary 4.4) and the completeness of SD (Theorem 3.1), both Π and $\overline{\Pi}$ reduce to SD. Let f and g be these reductions, respectively. Now, let $(\phi, x_1, \ldots, x_k)$ be any instance of $\Phi(\Pi)$, where $\phi = \phi(v_1, \ldots, v_k)$. Use De Morgan's laws to propagate all negations of ϕ to its variables. Now replace all occurrences of the literal $\neg v_i$ with a new variable w_i. Let $\psi(v_1, \ldots, v_k, w_1, \ldots, w_k)$ be the resulting (monotone) formula. It is clear that

$$(\phi, x_1, \ldots, x_k) \mapsto (\psi, f(x_1), \ldots, f(x_k), g(x_1), \ldots, g(x_k))$$

is a reduction from $\Phi(\Pi)$ to Mon(SD). Since Mon(SD) $\in$ SZK (Lemma 4.10) and SZK is closed under reductions (Corollary 4.3), Theorem 4.9 follows. ■

Theorem 4.9 can be also viewed as demonstrating that SZK is closed under a type of polynomial-time reducibility, which is formalized by the following two definitions.

Definition 4.12 (truth-table reduction [LLS75]): *We say a promise problem* Π truth-table reduces *to a promise problem* Γ *if there exists a (deterministic) polynomial-time computable function* f*, which on input* x *produces a tuple* $(y_1, \ldots, y_k)$ *and a boolean circuit* C *(with* k *input gates) such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow C(\chi_\Gamma(y_1), \ldots, \chi_\Gamma(y_k)) = 1 \\ x \in \Pi_N &\Rightarrow C(\chi_\Gamma(y_1), \ldots, \chi_\Gamma(y_k)) = 0 \end{aligned}$$

In other words, a truth-table reduction for promise problems is a nonadaptive Cook reduction which is allowed to make queries which violate the promise, but still must have an unambiguous output (in the strong sense formalized by Definition 4.6). We further consider the case where we restrict the complexity of computing the output of the reduction from the queries:

Definition 4.13 (NC^1 truth-table reductions): *A truth-table reduction* f *between promise problems is an* NC^1 truth-table reduction *if the circuit* C *produced by the reduction on input* x *has depth bounded by* $c_f \log |x|$*, where* c_f *is a constant independent of* x*.*

With these definitions, we can restate Theorem 4.9 as follows:

Corollary 4.14 SZK *is closed under* NC^1 *truth-table reductions.*

Proof: Any circuit of size s and depth d can be efficiently "unrolled" into a formula of size $2^d \cdot s$. Hence, an NC^1 truth-table reduction from Γ to Π gives rise to a Karp reduction from Γ to $\Phi(\Pi)$. Since SZK is closed under $\Phi(\cdot)$ and Karp reductions, it is also closed under NC^1 truth-table reductions. ■

It would be interesting to prove that SZK is closed under general truth-table reductions (or, even better, adaptive Cook reductions), or give evidence that this is not the case.

4.3 Knowledge complexity

Knowledge complexity [GMR89, GP99] is a generalization of zero knowledge which attempts to quantify how much a verifier learns from an interactive proof. A number of different measures have been proposed to accomplish this, most of which are based on the intuition that a verifier gains at most k bits of "knowledge" from an interaction if it can simulate the interaction with at most k bits of "help". Below we give terse definitions of the variants we consider. The first three definitions come from [GP99], and the last comes from [ABV95]. Let (P, V) be an interactive proof system for a promise problem Π. Then the knowledge complexity of (P, V) in various senses is defined as follows:

- **Hint sense:** We say that (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *hint* sense if there exists a PPT simulator S and a hint function $h : \Pi_Y \to \{0,1\}^*$ such that for all $x \in \Pi_Y$, $|h(x)| = k(|x|)$ and $\|S(x, h(x)) - \mathrm{View}_{P,V}(x)\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Strict oracle sense:** (P, V) is said to have perfect (resp., statistical) knowledge complexity $k(n)$ in the *strict oracle* sense if there exists a PPT oracle-machine S and an oracle $\mathcal{O}$ such that on every input $x \in \Pi_Y$, S queries $\mathcal{O}$ at most $k(|x|)$ times and $\left\| S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x) \right\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Oracle sense:** (P, V) is said to have perfect (resp., statistical) knowledge complexity $k(n)$ in the *oracle* sense if there exists a PPT oracle-machine S and an oracle $\mathcal{O}$ such that on every input $x \in \Pi_Y$, S queries $\mathcal{O}$ at most $k(|x|)$ times, S outputs '`fail`' with probability at most $1/2$, and $\left\| S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x) \right\|$ is 0 (resp., is bounded by a negligible function of $|x|$), where $S^{\mathcal{O}}(x)$ denotes the output distribution of S conditioned on non-failure.

- **Average oracle sense:** (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *average oracle* sense if there exists a PPT oracle-machine S and an oracle $\mathcal{O}$ such that for every input $x \in \Pi_Y$, the average number of queries S makes to $\mathcal{O}$ is at most $k(|x|)$ and $\left\| S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x) \right\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Entropy sense:** (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *entropy* sense if there exists a PPT oracle-machine S, an oracle $\mathcal{O}$, and a PPT oracle-simulator A such that for all $x \in \Pi_Y$, $\mathrm{E}_R[\log P_x(R)^{-1}] \leq k(|x|)$, where $P_x(R) = \Pr_\rho[A(x, R; \rho) = S^{\mathcal{O}}(x; R)]$ and $\left\| S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x) \right\|$ is 0 (resp., is bounded by a negligible function of $|x|$). Here, the notation $M(y; r)$ denotes the output of PPT M on input y and random coins r,

The *knowledge complexity* (in some specified sense) of a promise problem Π is $k(n)$ if there exists an interactive proof system (P, V) for Π achieving negligible error probablity in both the completeness and soundness conditions such that the knowledge complexity of (P, V) is $k(n)$. The class of languages possessing perfect knowledge complexity $k(n)$ in the hint, strict oracle, average oracle, and entropy senses are denoted by $\mathsf{PKC}_{\mathsf{hint}}$, $\mathsf{PKC}_{\mathsf{strict}}$, $\mathsf{PKC}_{\mathsf{avg}}$, and $\mathsf{PKC}_{\mathsf{ent}}$, respectively. Statistical knowledge complexity is denoted by SKC with the appropriate subscript.

A Collapse for the Hint Sense

Our first result about knowledge complexity is that the $\mathsf{SKC}_{\mathsf{hint}}$ hierarchy collapses by logarithmic additive factors. Previously, Goldreich and Petrank [GP99] have shown that $\mathsf{SKC}_{\mathsf{hint}}(\mathrm{poly}(n)) \subset \mathsf{AM}$ and $\mathsf{SKC}_{\mathsf{hint}}(O(\log(n))) \subset \mathsf{co\text{-}AM}$; the second of these results can be derived immediately from our result and Fortnow's theorem [For89] that $\mathsf{SZK} \subset \mathsf{co\text{-}AM}$.

Theorem 4.15 *For any polynomially bounded function $k(n)$,*

$$\mathsf{SKC}_{\mathsf{hint}}(k(n) + \log n) = \mathsf{SKC}_{\mathsf{hint}}(k(n)).$$

For intuition, consider the case that $k(n) = 0$. Loosely speaking, if the verifier is given the hint along with the input (with the "promise" that the hint is correct), then the original proof system becomes *zero* knowledge, so we can apply the results of the previous section. By the boolean closure properties established in Theorem 4.9, we can take the "union over all possible hints" (there are only polynomially many of them) without leaving SZK. The result is easily seen to be the original problem.

In order to turn this intuition into a proof, we first show that knowledge complexity in the hint sense can be characterized in terms of zero-knowledge promise problems, so that questions about the $\mathsf{SKC}_{\mathsf{hint}}$ hierarchy are reduced to questions about statistical zero knowledge. This is equivalence is obtained by providing the hint along with the input and "promising" that the hint is correct.

Lemma 4.16 *Let $k(n)$ be any polynomially bounded function. Then $\Pi \in \mathsf{SKC_{hint}}(k(n))$ (resp., $\mathsf{PKC_{hint}}(k(n))$) iff there exists a promise problem $\Gamma \in \mathsf{SZK}$ (resp., PZK) such that*

1. *$x \in \Pi_Y \Rightarrow$ there exists a such that $|a| = k(|x|)$ and $(x, a) \in \Gamma_Y$, and*
2. *$x \in \Pi_N \Rightarrow$ for all a, $(x, a) \in \Gamma_N$.*

Proof: We only give the proof for statistical knowledge complexity and zero knowledge; the perfect case is identical.

$\Rightarrow$ Let Π be a promise problem in $\mathsf{SKC_{hint}}(k(n))$ and let $h : \Pi_Y \to \{0,1\}^*$. be a hint function corresponding to an appropriate interactive proof system and simulator for Π. Consider the following promise problem Γ:

$$\begin{aligned} \Gamma_Y &= \{(x, h(x)) : x \in \Pi_Y\} \\ \Gamma_N &= \{(x, a) : x \in \Pi_N\} \end{aligned}$$

By using the protocol and simulator for Π, we see that $\Gamma \in \mathsf{SZK}$ (the verifier and prover for Γ should ignore the second component, whereas the simulator uses it as a hint.) It is clear that Γ satisfies the other conditions of Lemma 4.16.

$\Leftarrow$ Let $\Gamma \in \mathsf{SZK}$ be the promise problem satisfying the stated conditions. Let $h : \Pi_Y \to \{0,1\}^*$ be any function such that for all $x \in \Pi_Y$,

1. $|h(x)| = k(|x|)$,
2. $(x, h(x)) \in \Gamma_Y$.

(Such a function is guaranteed by Condition 1.) We now give a proof system for Π of knowledge complexity $k(n)$. On input x, the prover gives the verifier $h(x)$ in the first step, and then they execute the protocol for Γ on $(x, h(x))$. The completeness and soundness of this protocol follow from the properties of the Γ proof system. This proof system is easily seen to have knowledge complexity $k(n)$ in the hint sense, using the hint $h(x)$ with the the zero-knowledge simulator for Γ. ■

We now prove Theorem 4.15.

Proof: Let Π be a problem in $\mathsf{SKC_{hint}}(k(n) + \log n)$ and let Γ be the promise problem guaranteed by Lemma 4.16. By Theorem 4.9, $\Phi(\Gamma) \in \mathsf{SZK}$. Now consider a different, but related promise problem Γ', defined by

$$\begin{aligned} \Gamma'_Y &= \{(x, a)) : \text{there exists } b \text{ such that } |b| = \log|x| \text{ and } (x, ab) \in \Gamma_Y\} \\ \Gamma'_N &= \{(x, a) : \text{for all } b,\ (x, ab) \in \Gamma_N\} = \{(x, a) : x \in \Pi_N\}. \end{aligned}$$

For any string x, let $b_1, \ldots, b_n$ be all strings of length $\log|x|$, and let C be the circuit of depth $O(\log|x|)$ computing the function $\phi(v_1, \ldots, v_n) = \bigvee_i v_i$. The relationship between Γ and Γ' above implies that

$$(x, a) \mapsto (\phi, (x, ab_1), \ldots, (x, ab_n))$$

is an NC^1 truth-table reduction from Γ' to Γ. Since SZK is closed under such reductions (Corollary 4.14), we conclude that $\Gamma' \in \mathsf{SZK}$.

Now, $x \in \Pi_Y$, then there exists an a of length $k(|x|) + \log(|x|)$ such that $(x, a) \in \Gamma_Y$. Taking a' to be the first $k(|x|)$ bits of a, we see that there exists an a' of length $k(|x|)$ such that $(x, a') \in \Gamma'_Y$. Moreover, if $x \in \Pi_N$, then for all a, $(x, a) \in \Gamma'_N$. Thus, by Lemma 4.16, we conclude that $\Pi \in \mathsf{SKC}_{\mathsf{hint}}(k(n))$. ■

The Perfect Knowledge Complexity of SZK

The next theorem establishes tighter bounds on the perfect knowledge complexity of SZK. Aiello, Bellare, and Venkatesan [ABV95] have previously demonstrated that every language in SZK has perfect knowledge complexity $n^{-\omega(1)}$ (resp., $1 + n^{-\omega(1)}$) in the entropy (resp. average oracle) sense. Our results improve on these bounds, although the results of [ABV95] also apply to cheating-verifier classes and ours do not. Goldreich, Ostrovsky, and Petrank [GOP98] show that SZK has logarithmic perfect knowledge complexity in the oracle sense, so our results are incomparable to theirs. Our result for the strict oracle sense is the first that we know of.

Theorem 4.17 [11]

1. *For every polynomial-time computable* $m(n) = \omega(\log n)$, $\mathsf{SZK} \subset \mathsf{PKC}_{\mathsf{strict}}(m(n))$.
2. $\mathsf{SZK} \subset \mathsf{PKC}_{\mathsf{avg}}(1 + 2^{-n})$.
3. $\mathsf{SZK} = \mathsf{PKC}_{\mathsf{ent}}(2^{-n})$.

Corollary 4.2 tells us that every problem in SZK has a simple two-message proof system like the SD proof system of Section 3.3. Thus, in order to measure the perfect knowledge complexity of SZK and prove Theorem 4.17, it suffices to analyze this protocol. Intuitively, since the prover is only sending the verifier one bit and this bit is almost always a value the verifier knows, the knowledge complexity of this protocol should be extremely small. However, this argument does not suffice, because the knowledge complexity of a problem Π is determined only by proof systems for Π which achieve *negligible* error probability in both the completeness and soundness conditions. We can overcome this difficulty by performing $\omega(\log n)$ parallel repetitions.

Proof: Let Π be any problem in SZK and let (P, V) be the proof system for Π constructed in Corollary 4.2 (from the SD proof system of Section 3.3) with the security parameter set to $k = 4n$ (so the completeness error is 2^{-4n}). Let $m = m(n)$ be any function computable in time poly(n) such that $\omega(\log n) \le m \le n$. Consider the proof system (P', V') obtained by m parallel repetitions of (P, V); this has negligible completeness and soundness errors. We now analyze its perfect knowledge complexity.

1. The prover sends at most m bits to the verifier on inputs of length n, so the perfect knowledge complexity of this protocol in the strict oracle sense is bounded by m.
2. A perfect simulator for (P', V') can be obtained as follows: On input x of length n, the simulator runs $V(x)$ for m times independently and queries the oracle *once* to find out if

[11] The $2^{-\Omega(n)}$ in these results can be improved to $2^{-\Omega(n^k)}$ for any constant k by polarizing with security parameter n^k instead of $n+1$ in the SD proof system of Section 3.3.

any of these runs would result in an incorrect prover response. If the oracle replies yes, the simulator queries the oracle m more times to find out which runs would result in an incorrect response. The simulator then outputs the random coins used for running V and the appropriate prover responses.

In each subprotocol, the prover gives an incorrect response with probability at most 2^{-4n}. Thus, the simulator has to query the oracle for more than one bit with probability at most $n2^{-4n}$. Thus, on average, the simulator queries the oracle for at most $1 + mn2^{-4n} < 1 + 2^{-n}$ bits.

3. Let S be the simulator for (P', V') which simply simulates V' and queries the oracle $\mathcal{O}$ for all prover responses. One possible oracle simulator would assume that the prover is correct in all subprotocols. Unfortunately, this gives $1/P_x(R) = \infty$ for some R and yields infinite knowledge complexity. Thus, we instead have our oracle simulator A assume that the prover is right in each subprotocol independently with probability $1 - \delta$, where $\delta = 2^{-2n}$. Thus, $P_x(R) = (1-\delta)^k \delta^{m-k}$, if R is a set of random coins for V' (equivalently S, since S mimics V') which would elicit a correct prover response in exactly k of the subprotocols. Let ϵ be the probability that the prover is incorrect in an individual subprotocol. Then, $\epsilon \leq \delta^2$, and we have

$$\begin{aligned}
\mathrm{E}_R\left[\log \frac{1}{P_x(R)}\right] &= \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k \log\left(\frac{1}{(1-\delta)^k \delta^{m-k}}\right) \\
&= \left(\log \frac{1}{\delta^m}\right) \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k + \left(\log \frac{\delta}{1-\delta}\right) \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k k \\
&= \log \frac{1}{\delta^m} + m(1-\epsilon)\left(\log \frac{\delta}{1-\delta}\right) \\
&= m\left(\log \frac{1}{1-\delta} + \epsilon \log \frac{1-\delta}{\delta}\right) \\
&\leq m\left(\log \frac{1}{1-\delta} + \delta^2 \log \frac{1}{\delta}\right) \\
&\leq 2m\delta < 2^{-n}
\end{aligned}$$

for sufficiently large n.

The opposite inclusion follows from the result of [ABV95] that $\mathsf{PKC}_{\mathsf{ent}}(\mathrm{neg}(n)) \subset \mathsf{SZK}$ for any negligible function $\mathrm{neg}(n)$.

■

4.4 Reversing statistical difference

By the completeness of SD (Theorem 3.1) and SZK's closure under complement (Corollary 4.4), it follows that $\overline{\mathrm{SD}}$ reduces to SD. This is equivalent to the following surprising result:

Corollary 4.18 (Reversal Mapping) *There is a polynomial-time computable function that maps pairs of circuits (C_0, C_1) to pairs of circuits (D_0, D_1) such that*

$$\begin{aligned}
\|C_0 - C_1\| < 1/3 \quad &\Rightarrow \quad \|D_0 - D_1\| > 2/3 \\
\|C_0 - C_1\| > 2/3 \quad &\Rightarrow \quad \|D_0 - D_1\| < 1/3.
\end{aligned}$$

That is, SD *reduces to* $\overline{\text{SD}}$.

This corollary motivated our search for a more explicit description of such a mapping. By extracting ideas used in the transformations of statistical zero-knowledge proofs given in [Oka00] and [SV97], we obtained the description of this transformation given below.

The Construction. Let (C_0, C_1) be any pair of circuits and let $n = |(C_0, C_1)|$. By the Polarization Lemma (Lemma 3.3), we can produce in polynomial time a pair of circuits (C_0', C_1') such that

$$\begin{aligned} \|C_0 - C_1\| < 1/3 \quad &\Rightarrow \quad \left\|C_0' - C_1'\right\| > 1 - 2^{-n} \\ \|C_0 - C_1\| > 2/3 \quad &\Rightarrow \quad \left\|C_0' - C_1'\right\| < 2^{-n} \end{aligned}$$

Let $q = \text{poly}(n)$ be the number of input gates of C_0' and C_1' (w.l.o.g. we may assume they have the same number) and let $\ell = \text{poly}(n)$ be the number of output gates. For notational convenience, let $R = \{0,1\}^q$ and $L = \{0,1\}^\ell$. Let $m = n^3 q^2$ and define a new distribution $\vec{C}\colon \{0,1\}^m \times R^m \to L^m$ as follows:

$$\vec{C}(\vec{b}, \vec{r}) = (C'_{b_1}(r_1), \ldots, C'_{b_m}(r_m)).$$

We use the notation $\vec{z} \leftarrow \vec{C}$ to denote $\vec{z}$ chosen according to $\vec{C}$, *i.e.* select $\vec{b}$ and $\vec{r}$ uniformly and let $\vec{z} = \vec{C}(\vec{b}, \vec{r})$.

Let $\mathcal{H}$ be a 2-universal family of hash functions from $\{0,1\}^m \times R^m \times L^m$ to $T = \{0,1\}^{(q+1)m - 2\Delta - n}$, where $\Delta = \sqrt{nmq^2} = m/n$. We can now describe the new distributions:

D_0: Choose $(\vec{b}, \vec{r}) \in_R \{0,1\}^m \times R^m$, $\vec{y} \leftarrow \vec{C}$, and $h \in_R \mathcal{H}$. Output $(\vec{C}(\vec{b}, \vec{r}), \vec{b}, h, h(\vec{b}, \vec{r}, \vec{y}))$.
D_1: Choose $(\vec{b}, \vec{r}) \in_R \{0,1\}^m \times R^m$, $h \in_R \mathcal{H}$, and $t \in_R T$. Output $(\vec{C}(\vec{b}, \vec{r}), \vec{b}, h, t)$.

The important things to note about these distributions are that $\vec{b}$ is part of the output, and that D_0 and D_1 only differ in the last component, where D_0 has the value of the hash function and D_1 has a truly random element of T. Also note that the size of T is chosen to be $|\{0,1\}^m \times R^m|/2^{2\Delta+n}$, which is essentially $|\{0,1\}^m \times R^m|$, scaled down by a "slackness" factor of $2^{2\Delta+n}$. The introduction of the sample $\vec{y}$ in D_0 may at first seem superfluous; we explain below.

Intuition. For intuition, consider the case that $\vec{C}$ is a *flat* distribution; that is, for every $\vec{z} \in \text{range}(\vec{C})$, the size of the preimage set $|\{(\vec{b}, \vec{r})\colon \vec{C}(\vec{b}, \vec{r}) = \vec{z}\}|$ is the same value N. (It turns out that $\vec{C}$ is actually "close enough" to being flat for the following arguments to work.) Then the range of $\vec{C}$ has size $|\{0,1\}^m \times R^m|/N = 2^{(q+1)m}/N$. So, in D_0, conditioned on a value for $\vec{C}(\vec{b}, \vec{r})$, the triple $(\vec{b}, \vec{r}, \vec{y})$ is selected uniformly from a set of size $2^{(q+1)m}$. Since this is much greater than $|T|$, the Leftover Hash Lemma of [HILL99] implies that conditioned on any value for the first component of D_0, the last two components $(h, h(\vec{b}, \vec{r}, \vec{y}))$ are distributed close to the uniform distribution on $\mathcal{H} \times T$, which is the distribution that D_1 has in its last two components.[12] Thus, if their second

[12]Here we see the importance of $\vec{y}$: Without $\vec{y}$, conditioned on some value of $\vec{C}(\vec{b}, \vec{r})$, the pair $(\vec{b}, \vec{r})$ would be selected uniformly from a space of size N. If we were only hashing this pair, for the distribution $h(\vec{b}, \vec{r})$ to be uniform by the Leftover Hash Lemma, T would have had to be chosen so that $|T| \ll N$. The value of N, however, depends on the inner workings of the circuit C, and is in general unknown. By including $\vec{y}$, which comes uniformly from a space of size $2^{(q+1)m}/N$, we balance the arguments to h so that they come from a space of size $2^{(q+1)m}$, a known quantity. This use of "dummy" samples to form a space whose size is known is the "complementary usage of messages" technique of Okamoto [Oka00].

components were missing, D_0 and D_1 would be statistically close. Now, consider the case that $\|C'_0 - C'_1\| \approx 1$. Then $\vec{b}$ is essentially "determined" by $\vec{C}(\vec{b},\vec{r})$. So the presence of $\vec{b}$ can be ignored, and the above argument says that D_0 and D_1 are statistically very close. Now, consider the case that $\|C'_0 - C'_1\| \approx 0$. Then $\vec{b}$ is essentially "unrestricted" by $\vec{C}(\vec{b},\vec{r})$. Since there are 2^m choices for $\vec{b}$, conditioning on $\vec{b}$ in addition to $\vec{C}(\vec{b},\vec{r})$, cuts the number of triples $(\vec{b},\vec{r},\vec{y})$ down from $2^{m(q+1)}$ to roughly $2^{m(q+1)}/2^m$. Since $2^{m(q+1)}/2^m$ is much smaller than $|T|$, $h(\vec{b},\vec{r},\vec{y})$ will cover only a small fraction of $|T|$ and thus will be far from uniform (conditioned on values for $\vec{C}(\vec{b},\vec{r})$, $\vec{b}$, and h).

Direct Proof of Corollary 4.18. First we will argue that $\vec{C}$ is close to being flat, so that we can apply arguments like those given above. This is the case because $\vec{C}$ is composed of many independent, identically distributed random variables. For $\vec{z} \in L^m$, we say the weight of $\vec{z}$ is the logarithm of the size of the preimage set of $\vec{z}$. Formally, let $\mathrm{wt}(\vec{z}) = \log_2 |\{(\vec{b},\vec{r}) : \vec{C}(\vec{b},\vec{r}) = \vec{z}\}|$. Let w be the expected weight of an image, i.e. $w = \mathrm{E}_{\vec{z}\leftarrow\vec{C}}[\mathrm{wt}(\vec{z})]$. Then we can show the following:

Lemma 4.19 $\Pr_{\vec{z}\leftarrow\vec{C}}[|\mathrm{wt}(\vec{z}) - w| > \Delta] < 2^{-\Omega(n)}$.

Proof: For $z \in L$, let $\mathrm{wt}_0(z) = \log_2 |\{(b,r) : C_b(r) = z\}|$. Then, for $\vec{z} \in L^m$, $\mathrm{wt}(\vec{z}) = \mathrm{wt}_0(z_1) + \cdots + \mathrm{wt}_0(z_m)$. Observe that when $\vec{z}$ is selected according to $\vec{C}$, $z_1, \ldots, z_m$ are independent and identically distributed. Moreover, for any $z \in L$, $0 \le \mathrm{wt}_0(z) \le q$. So, by the Hoeffding inequality [Hof95, Sec. 7.2.1], we have

$$\Pr_{\vec{z}\leftarrow\vec{C}}[|\mathrm{wt}(\vec{z}) - w| > \Delta] < 2e^{-2\Delta^2/mq^2} = 2e^{-2n}.$$

■

It will be convenient to eliminate those $\vec{z} \in L^m$ that have weight far above or below the mean. Let $G = \{(\vec{b},\vec{r}) : |\mathrm{wt}(\vec{C}(\vec{b},\vec{r})) - w| \le \Delta\}$ be the set of *good* pairs $(\vec{b},\vec{r})$. The above Lemma says that $|G| \ge (1 - 2^{-\Omega(n)})|\{0,1\}^m \times R^m|$. Thus $\|G - \{0,1\}^m \times R^m\| \le 2^{-\Omega(n)}$, where for simplicity of notation, we let the name of a set also refer to the uniform distribution on the same set. Define $\vec{C}'$ to be the distribution obtained by selecting $(\vec{b},\vec{r}) \leftarrow G$ and outputting $\vec{C}(\vec{b},\vec{r})$. Then, since $\vec{C}$ is a function, Fact 2.4 tells us that $\|\vec{C} - \vec{C}'\| = 2^{-\Omega(n)}$. Similarly, we define variants of D_0 and D_1 that sample from G instead of $\{0,1\}^m \times R^m$:

D'_0: Let $(\vec{b},\vec{r}) \in_R G$, $\vec{y} \leftarrow \vec{C}'$, and $h \in_R \mathcal{H}$. Output $(\vec{C}'(\vec{b},\vec{r}), \vec{b}, h, h(\vec{b},\vec{r},\vec{y}))$.
D'_1: Let $(\vec{b},\vec{r}) \in_R G$, $h \in_R \mathcal{H}$, and $t \in_R T$. Output $(\vec{C}'(\vec{b},\vec{r}), \vec{b}, h, t)$.

Since D'_0 (or D'_1) is a randomized procedure applied to two (or one) independent samplings from G, Fact 2.4 tells us that $\|D_0 - D'_0\| = 2^{-\Omega(n)}$ (and $\|D_1 - D'_1\| = 2^{-\Omega(n)}$). Hence, it suffices to prove that these modified distributions have the properties we want in each case. For the case when C_0 and C_1 are statistically far, we prove the following claim:

Claim 4.20 *If $\|C'_0 - C'_1\| > 1 - 2^{-n}$, then $\|D'_0 - D'_1\| < 2^{-\Omega(n)}$.*

Proof of claim: First we formalize the idea that $\vec{b}$ is "determined" by $\vec{C}$. Define $f : L \to \{0,1\}$ by

$$f(z) = \begin{cases} 0 & \text{if } \Pr[C'_0 = z] > \Pr[C'_1 = z] \\ 1 & \text{otherwise} \end{cases}$$

In other words, f is exactly the prover strategy from the proof system for STATISTICAL DIFFERENCE given in Section 3.3. The completeness of that proof system (Lemma 3.7) says that $\Pr_{b,r}[f(C'_b(r)) = b] > 1 - 2^{-n}$. Now define $\vec{f} : L^m \to \{0,1\}^m$ by $\vec{f}(\vec{z}) = (f(z_1), \ldots, f(z_m))$. Then

$$\Pr_{\vec{b},\vec{r}}[\vec{f}(\vec{C}(\vec{b},\vec{r})) = \vec{b}] > (1 - 2^{-n})^m = 1 - 2^{-\Omega(n)}.$$

Since G is a $1 - 2^{-\Omega(n)}$ fraction of $\{0,1\}^m \times R^m$, the same is true when $(\vec{b},\vec{r})$ is selected uniformly from G. Thus, if we define:

D''_0: Let $(\vec{b},\vec{r}) \in_R G$, $\vec{y} \leftarrow \vec{C}'$, and $h \in_R \mathcal{H}$. Output $(\vec{C}'(\vec{b},\vec{r}), \vec{f}(\vec{C}'(\vec{b},\vec{r})), h, h(\vec{b},\vec{r},\vec{y}))$.

D''_1: Let $(\vec{b},\vec{r}) \in_R G$, $h \in_R \mathcal{H}$, and $t \in_R T$. Output $(\vec{C}'(\vec{b},\vec{r}), \vec{f}(\vec{C}'(\vec{b},\vec{r})), h, t)$.

Then, by Fact 2.5, $\|D'_0 - D''_0\| = 2^{-\Omega(n)}$ and $\|D'_1 - D''_1\| = 2^{-\Omega(n)}$. So it suffices to show that $\|D''_0 - D''_1\| = 2^{-\Omega(n)}$. Since the first components of D''_0 and D''_1 are identically distributed and the second components are determined by the first ones, it suffices to show (by Fact 2.5) that, conditioned on any value for the first coordinate, the third and fourth components have statistical difference $2^{-\Omega(n)}$. This will follow from the Leftover Hash Lemma [HILL99]:

Lemma 4.21 (Leftover Hash Lemma [HILL99]) *Let $\mathcal{H}$ be a family of 2-universal hash functions from D to T. Let X by a probability distribution on D such that for all $x \in D$, $\Pr[X = x] \le \epsilon/|T|$. Then the following two distributions have statistical difference at most $\epsilon^{1/3}$.*

1. *Choose $x \leftarrow X$, $h \in_R \mathcal{H}$. Output $(h, h(x))$.*
2. *Choose $h \in_R \mathcal{H}$, $t \in_R T$. Output (h, t).*

By the above argument and the Leftover Hash Lemma, it suffices to show that conditioned on any value $\vec{z}$ for $\vec{C}'(\vec{b},\vec{r})$, no triple $(\vec{b},\vec{r},\vec{y})$ has probability more than $2^{-\Omega(n)}/|T|$. The pair $(\vec{b},\vec{r})$ comes uniformly from a set of size $2^{\mathrm{wt}(\vec{z})} \ge 2^{w-\Delta}$, and $\vec{y}$ is selected independently according to $\vec{C}'$, so the probability of any triple $(\vec{b},\vec{r},\vec{y})$ is at most

$$\left(\frac{1}{2^{w-\Delta}}\right)\left(\frac{2^{w+\Delta}}{|G|}\right) \le \frac{2^{2\Delta}}{(1 - 2^{-\Omega(n)})2^{(q+1)m}} = \frac{2^{-\Omega(n)}}{|T|}.$$

Thus, $\|D''_0 - D''_1\| \le 2^{-\Omega(n)}$, and the claim is established.

Now we treat the other case, when C_0 and C_1 are statistically close.

Claim 4.22 *If $\|C'_0 - C'_1\| < 2^{-n}$, then $\|D'_0 - D'_1\| > 1 - 2^{-\Omega(n)}$.*

Proof of claim: First, we formalize the idea that $\vec{b}$ is almost completely "undetermined" by $\vec{C}(\vec{b},\vec{r})$. Since $\|C'_0 - C'_1\| < 2^{-n}$, it follows from Fact 2.6 that with probability $1 - 2^{-\Omega(n)}$ over $z \leftarrow C'_0$,

$$(1 - 2^{-\Omega(n)}) \Pr\left[C'_1 = z\right] \le \Pr\left[C'_0 = z\right] \le (1 + 2^{-\Omega(n)}) \Pr\left[C'_1 = z\right].$$

In other words,

$$1 - 2^{-\Omega(n)} \leq \frac{|\{r : C_0'(r) = z\}|}{|\{r : C_1'(r) = z\}|} \leq 1 + 2^{-\Omega(n)}.$$

The same is true with probability $1 - 2^{-\Omega(n)}$ when the roles of C_0' and C_1' are reversed. Thus, with probability $1 - m2^{-\Omega(n)} = 1 - 2^{-\Omega(n)}$ over $\vec{z} \leftarrow \vec{C}$, we have for *every* pair $\vec{b}, \vec{c} \in \{0,1\}^m$,

$$1 - 2^{-\Omega(n)} = (1 - 2^{\Omega(n)})^m \leq \frac{\left|\{\vec{r} : \vec{C}(\vec{b}, \vec{r}) = \vec{z}\}\right|}{\left|\{\vec{r} : \vec{C}(\vec{c}, \vec{r}) = \vec{z}\}\right|} \leq (1 + 2^{-\Omega(n)})^m = 1 + 2^{-\Omega(n)}.$$

Since there are 2^m choices for $\vec{c}$, this, combined with Lemma 4.19, implies that, with probability $1 - 2^{-\Omega(n)}$ over $\vec{z} \leftarrow \vec{C}$, the following holds for *every* $\vec{b} \in \{0,1\}^m$:

$$\left|\{\vec{r} : \vec{C}(\vec{b}, \vec{r}) = \vec{z}\}\right| \leq (1 + 2^{-\Omega(n)}) \cdot \frac{2^{\mathrm{wt}(\vec{z})}}{2^m} \leq (1 + 2^{-\Omega(n)}) \cdot 2^{w + \Delta - m}.$$

Since this is true with probability $1 - 2^{-\Omega(n)}$ for $\vec{z}$ selected according to $\vec{C}$, it is also true with probability $1 - 2^{-\Omega(n)}$ for $\vec{z}$ selected according to $\vec{C}'$. Fix any such $\vec{z}$ and fix any $\vec{b} \in \{0,1\}^m$ and $h \in \mathcal{H}$. Then, in D_0', conditioned on $\vec{C}'(\vec{b}, \vec{r}) = \vec{z}$, $\vec{b}$, and h, there are at most

$$\begin{aligned}
(1 + 2^{-\Omega(n)}) \cdot 2^{w + \Delta - m} \left(\frac{|G|}{2^{w - \Delta}}\right) &\leq (1 + 2^{-\Omega(n)}) \cdot 2^{2\Delta - m} \cdot 2^{m(q+1)} \\
&= (1 + 2^{-\Omega(n)}) \cdot 2^{4\Delta + n - m} \cdot |T| \\
&= 2^{-\Omega(m)} \cdot |T|
\end{aligned}$$

possible values for $(\vec{r}, \vec{y})$. Thus, with probability $1 - 2^{-\Omega(n)}$, conditioned on values for the first three components of D_0', the fourth component $h(\vec{b}, \vec{r}, \vec{y})$ can cover at most a $2^{-\Omega(m)} \leq 2^{-\Omega(n)}$ fraction of T. In contrast, conditioned on values for the first three components of D_1', the fourth component is uniformly distributed on T. Therefore, $\|D_0' - D_1'\| \geq 1 - 2^{-\Omega(n)}$.

In [Vad99], it is shown that this Reversal Mapping can be better understood as a composition of two reductions, going the two directions between Statistical Difference and Entropy Difference (the complete problem for SZK given in [GV99], which trivially reduces to its complement).

4.5 Weak-SZK and expected polynomial-time simulators

Recall that, in this paper, we defined statistical zero-knowledge with respect to *strict* polynomial-time simulators. As noted in Section 2, the original definition of statistical zero-knowledge permits *expected* polynomial-time simulators, but only allowing strict polynomial-time simulators is not very restrictive when discussing honest-verifier proofs, as we are.

However, our techniques do say something about expected polynomial-time simulators, and in particular show that expected polynomial-time simulators are no more powerful than strict ones for public-coin statistical zero-knowledge. This is the first general equivalence between strict and expected polynomial-time simulators for statistical zero knowledge that we know of.

Indeed, we are able to generalize further to an even weaker notion, that of *weak* statistical zero knowledge (as previously considered in [DOY97], where it was referred to as "non-uniform simulation"):

Definition 4.23 *An interactive proof system (P, V) for a promise problem Π is* weak statistical zero knowledge *if for all polynomials p, there exists an efficient probabilistic (strict) polynomial-time algorithm S_p such that*

$$\|S_c(x) - (P, V)(x)\| \leq 1/p(|x|),$$

for all sufficiently long $x \in \Pi_Y$.

We denote by weak-SZK the class of promise problems admitting weak statistical zero-knowledge proofs, and by public-coin weak-SZK the class corresponding to such proofs which are also public coin. Note that any proof system admitting an expected polynomial-time simulator (in the usual sense) certainly also satisfies the requirements of weak statistical zero-knowledge. We show that in fact any public-coin weak statistical zero-knowledge proof system can be transformed into a statistical zero-knowledge proof system with a strict polynomial-time simulator achieving negligible (in fact, exponentially small) simulator deviation. In other words, public-coin weak-SZK = SZK.

Proposition 4.24 public-coin weak-SZK = SZK = public-coin SZK.

The only obstacle in generalizing Proposition 4.24 to all weak statistical zero-knowledge proofs (instead of just public-coin ones) is that Okamoto's private to public-coin transformation in [Oka00] is only given for strict polynomial-time simulators achieving negligible simulator deviation. In fact, this generalization was accomplished in work (subsequent to ours) by Goldreich and Vadhan [GV99].

In order to establish Proposition 4.24, it suffices to show that every problem in public-coin weak-SZK reduces to SD, as the proposition follows by closure under reductions (Corollary 4.3) and Okamoto's theorem that SZK = public-coin SZK (Theorem 3.10). Therefore, we need only establish the following generalization of Lemma 3.8:

Lemma 4.25 *Suppose promise problem Π has a public-coin weak statistical zero-knowledge proof. Then there exist probabilistic (strict) polynomial time machines A and B such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow \|A(x) - B(x)\| < \frac{1}{3}, \text{ and} \\ x \in \Pi_N &\Rightarrow \|A(x) - B(x)\| > \frac{2}{3}. \end{aligned}$$

Proof: The proof is identical to the proof of Lemma 3.8, except that wherever the simulator S is used in that proof, we replace it with S_p, a simulator with deviation $1/p(n)$, where $p(n) = 7n \cdot r(n)^3$. Then we replace Claim 3.12 with the following:

Claim 4.26 *If $x \in \Pi_Y$, then $\|A(x) - B(x)\| \leq 1/(3|x| \cdot r(|x|)^2)$.*

> **Proof of claim:** The proof is identical to the proof of Claim 3.12, except that now, we have
>
> $$\|A(x) - B(x)\| \leq 2^{-\Omega(|x|)} + 2r(|x|) \cdot \|S_p(x) - (P, V)(x)\| < \frac{1}{3|x| \cdot r(|x|^2}.$$

On the other hand, Claim 3.13 remains true, i.e. $x \in \Pi_N$ implies $\|A(x) - B(x)\| \geq 1/12r(n)$. Then, as in the original proof, we consider the samplable distributions $\hat{A}(x) = \otimes^{s(|x|)} A(x)$ and $\hat{B}(x) = \otimes^{s(|x|)} B(x)$, where $s(n) = n \cdot r(n)^2$. If $x \in \Pi_Y$, $\|\hat{A}(x) - \hat{B}(x)\| \leq s(|x|)\|A(x) - B(x)\| < 1/3$, as desired. If $x \in \Pi_N$, then by the Direct Product Lemma (Lemma 3.4), $\|\hat{A}(x) - \hat{B}(x)\| \geq 1 - 2^{-\Omega(|x|)}$. ■

4.6 Perfect and computational zero knowledge

Although the focus of this paper is statistical zero knowledge, some of the techniques also apply to perfect and computational zero knowledge. In particular, for public-coin proof systems we obtain variants of Lemma 3.8 for both perfect and computational zero knowledge. In addition, a restricted version of STATISTICAL DIFFERENCE can be shown to have perfect zero-knowledge proof.

First, we define some variants of SD. For any two constants α and β with $\alpha > \beta$, define:

$$\begin{aligned} \mathrm{SD}_Y^{\alpha,\beta} &= \{(C_0, C_1) : \|C_0 - C_1\| \geq \alpha\} \\ \mathrm{SD}_N^{\alpha,\beta} &= \{(C_0, C_1) : \|C_0 - C_1\| \leq \beta\} \end{aligned}$$

$\mathrm{SD}^{\alpha,\beta}$ is interreducible with SD and hence complete for SZK whenever $1 > \alpha^2 > \beta > 0$, since the Polarization Lemma generalizes to such thresholds. (See discussion at the end of Section 3.2).

We can almost show that every problem which has a public-coin perfect zero-knowledge proof reduces to $\overline{\mathrm{SD}^{1/2,0}}$. The caveats are that either the original proof system must have perfect completeness, or we obtain distributions that are samplable in *expected* polynomial time rather than circuits.

Proposition 4.27 *Every promise problem having a public-coin perfect zero-knowledge proof with perfect completeness reduces to* $\overline{\mathrm{SD}^{1/2,0}}$.

Proof: It suffices to show that the distributions $A(x)$ and $B(x)$ constructed in the proof of Lemma 3.8 have statistical difference 0 on YES instances, when the original proof system has perfect completeness and the simulator deviation is 0. Indeed, for $i \geq 1$, the distributions $A_i(x)$ and $B_i(x)$ are identical if the simulator deviation is 0, and the distributions $A_0(x)$ and $B_0(x)$ are identical under the additional assumption that the proof system has perfect completeness. ■

Proposition 4.28 *Suppose promise problem* Π *has a public-coin perfect zero-knowledge proof. Then there exist probabilistic* expected *polynomial time machines* A *and* B *such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow \|A(x) - B(x)\| = 0, \text{ and} \\ x \in \Pi_N &\Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}. \end{aligned}$$

Proof: The proof is nearly identical to that of Proposition 4.27, except that we must modify $A_0(x)$ and $B_0(x)$ to have statistical difference 0 (at the price of $B_0(x)$ becoming expected polynomial time). Let $c(n)$ be a polynomial bound on the number of random coins S uses on inputs of length n. Then we define A_0 and B_0 as follows (in both descriptions, $n = |x|$):

$A_0(x)$: Run $S(x)$ for $n \cdot c(n)$ repetitions. Output '1' if the majority are accepting conversations and '0' otherwise.

$B_0(x)$: With probability $1 - 2^{-c(n)}$, output '1'. Otherwise, calculate the probability σ that $S(x)$ outputs an accepting conversation (by exhaustive search over all $2^{c(n)}$ random seeds). Now calculate

$$\tau = \sum_{i=0}^{\lfloor \frac{nc(n)}{2} \rfloor} \binom{nc(n)}{i} \sigma^i (1-\sigma)^{nc(n)-i}.$$

If $\tau > 2^{-c(n)}$, output '1.' Otherwise, output '0' with probability $\tau/2^{-c(n)}$, and '1' otherwise.

Note that $B_0(x)$ runs in expected polynomial time, since with probability $2^{-c(n)}$ it runs in time $\mathrm{poly}(n)2^{c(n)}$ and otherwise it runs in time $\mathrm{poly}(n)$. Also observe that τ is the probability that $A_0(x)$ outputs '0'.

Now we argue that, when $x \in \Pi_Y$, $A_0(x)$ and $B_0(x)$ have statistical difference 0,*i.e.* output '1' with the same probability. Since $S(x)$ outputs a conversation which makes V accept with probability at least $2/3 - \mathrm{neg}(n)$, the Chernoff bound implies that $\Pr[A_0(x) = 1] = 1 - 2^{-\Omega(nc(n))}$. This means that τ will always be less than $2^{-c(n)}$ (for sufficiently large n), so B_0 will output '0' with probability $2^{-c(n)}(\tau/2^{-c(n)}) = \tau$, which is the probability that A_0 outputs '0'. ■

Now, if we could show that $\mathrm{SD}^{1/2,0}$ (or its complement) has a perfect zero-knowledge proof system, we would have something like a completeness result for PZK. Although we do not know how to do this, we can instead show that $\mathrm{SD}^{1,1/2} \in \mathsf{PZK}$. Indeed, consider the protocol of Section 3.3 with the modification that the two parties use the XOR Lemma (Lemma 3.5) instead of the Polarization Lemma. Then the proof of Lemma 3.7 tells us that this protocol, when used for $\mathrm{SD}^{1,1/2}$ has completeness error 0, simulator deviation 0, and soundness error $1/2 + 2^{-n}$. Thus we have:

Proposition 4.29 $\mathrm{SD}^{1,1/2} \in \mathsf{PZK}$.

For *computational* zero knowledge, the techniques of Lemma 3.8 give us the following:

Proposition 4.30 *Suppose promise problem Π has a public-coin computational zero-knowledge proof. Then there exist probabilistic polynomial time machines A and B such that*

1. $x \in \Pi_N \Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}$, *and*
2. $\{A(x)\}_{x\in\Pi_Y}$ *and* $\{B(x)\}_{x\in\Pi_Y}$ *are* computationally indistinguishable *ensembles of probability distributions.*

Note that, in contrast to perfect and statistical zero knowledge, the conditions given in Proposition 4.30 do not give a way to distinguish YES and NO instances; it is possible for $A(x)$ and $B(x)$ to have statistical difference greater than $1 - 2^{-\Omega(|x|)}$ even for $x \in \Pi_Y$. We also remark that Proposition 4.30 holds even when the simulator for the proof system runs in expected polynomial-time, except that A and B will also run in expected polynomial-time.

Proof: The proof follows Lemma 3.8 exactly, except for Claim 3.12, which should be replaced with the following:

Claim 4.31 $\{A(x)\}_{x\in\Pi_Y}$ *and* $\{B(x)\}_{x\in\Pi_Y}$ *are computationally indistinguishable ensembles of probability distributions.*

We omit x from the notation for readability; below all probability distributions actually refer to *ensembles* indexed by $x \in \Pi_Y$. The proof in Claim 3.12 that A_0 and B_0 have exponentially small statistical difference still holds. Thus it suffices to show that the distributions A' and B' obtained by removing the 0'th components of A and B, respectively, are computationally indistinguishable.

To prove this, we first note that a hybrid argument shows that the distributions $\otimes^r(P,V)$ and $\otimes^r S$ are computationally indistinguishable, since (P,V) and S are computationally indistinguishable.[13]

Now we introduce a new distribution C. Define $C_i = (c_1, p_1, \ldots, c_i)_{(P,V)}$ for $1 \le i \le r$, and let $C = C_1 \otimes \cdots \otimes C_r$. Then C and A' are computationally indistinguishable since a distinguisher D between them could be used to make a distinguisher D' between $\otimes^r(P,V)$ and $\otimes^r S$: Given a sequence of r transcripts $(t_1, \ldots, t_r)$, D' truncates $t_i = (c_1, p_1, \ldots, c_r, p_r)$ to produce $t'_i = (c_1, p_1, \ldots, c_i)$ and feeds $(t'_1, \ldots, t'_r)$ to D. When fed with $\otimes^r S$, D' gives D a sample of A', and when fed with $\otimes^r(P,V)$, D' gives D a sample of C.

Similarly, C and B' are also computationally indistinguishable because a distinguisher between them could be to make a distinguisher D' between $\otimes^r(P,V)$ and $\otimes^r S$: Given a sequence of r transcripts $(t_1, \ldots, t_r)$, D' truncates $t_i = (c_1, p_1, \ldots, c_r, p_r)$ and selects u_i according to the uniform distribution on strings of length $r(|x|)$ to produce $t'_i = (c_1, p_1, \ldots, p_{i-1}, u)$ and feeds $(t'_1, \ldots, t'_r)$ to D. When fed with $\otimes^r S$, D' gives D a sample of B', and when fed with $\otimes^r(P,V)$, D' gives D a sample of C.

Now, because both A' and B' are computationally indistinguishable from C, they must be computationally indistinguishable from each other, completing the proof. ■

4.7 Hard-on-average problems and one-way functions

Most, if not all, of cryptography relies on the existence of computational problems which are hard-on-average. However, the mere exitence of a hard-on-average problem, even in NP, is not known to imply even the most basic cryptographic primitive, namely a one-way function. Ostrovsky [Ost91], however, showed that the existence of a hard-on-average problem in SZK *does* imply the existence of one-way functions. This result was subsequently generalized to CZK by Ostrovsky and Wigderson [OW93].

In this section, we show how Ostrovsky's result follows readily from our Completeness Theorem and a result of Goldreich [Gol90] on computational indistinguishability. Using the generalization of our techniques to CZK described in the previous section, we also obtain a simpler proof of the the Ostrovksy–Wigderson theorem restricted to public-coin proof systems.

In order to state these theorems precisely, we need to define what we mean for a problem Π to be "hard." Informally, we require that membership in Π is (very) hard to decide under some samplable distribution of instances.

Definition 4.32 *An ensemble of distributions $\{D_n\}_{n\in\mathbb{N}}$ is said to be* samplable *if there is a probabilistic polynomial-time algorithm that, on input 1^n outputs a string distributed according to D_n.*

Definition 4.33 *A promise problem Π is* hard-on-average *if there exists a samplable ensemble of distributions $\{D_n\}_{n\in\mathbb{N}}$ such that the following holds: For every nonuniform probabilistic polynomial-time algorithm M, there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that*

$$\Pr[M(x) \text{ correctly decides whether } x \text{ is a YES or NO instance of } \Pi] \le \frac{1}{2} + \mu(n) \qquad \forall n \in \mathbb{N},$$

where the probability is taken over $x \leftarrow D_n$ and the coins of M. (If x violates the promise, then M is considered to be correct no matter what it outputs.)

[13] Actually this step uses the fact that our definition of computational indistinguishability is with respect to nonuniform distinguishers, because (P,V) is not a samplable distribution.

In this section, we will give new proofs of the following results.

Theorem 4.34 ([Ost91]) *If there is a hard-on-average promise problem in* SZK*, then one-way functions exist.*

Theorem 4.35 ([OW93] for public-coin proofs) *If a hard-on-average promise problem possesses a public-coin computational zero-knowledge proof system, then one-way functions exist.*

We will only prove Theorem 4.35 as Theorem 4.34 then follows via Theorem 3.10. Our proof will make use of Proposition 4.30 in conjuction with the following result of Goldreich [Gol90]:

Proposition 4.36 ([Gol90]) *Suppose there exist two samplable ensembles of distributions,* $\{A_n\}_{n\in\mathbb{N}}$ *and* $\{B_n\}_{n\in\mathbb{N}}$*, such that*

1. $\{A_n\}$ *and* $\{B_n\}$ *are computationally indistinguishable.*
2. *There is a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for all* n, $\|A_n - B_n\| \geq 1/p(n)$.

Then one-way functions exist.

Proof of Theorem 4.34: Suppose Π is a hard-on-average problem with a public-coin computational zero-knowledge proof and let $\{D_n\}$ be the ensemble of distributions under which Π is hard. By Proposition 4.30 there are probabilistic polynomial-time algorithms A and B such that

1. $x \in \Pi_N \Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}$, and
2. $\{A(x)\}_{x\in\Pi_Y}$ and $\{B(x)\}_{x\in\Pi_Y}$ are computationally indistinguishable.

(Note that if $\Pi \in \mathsf{SZK}$, the Completeness Theorem and Polarization Lemma yield such A and B with the computational indistinguishability replaced by statistical difference $2^{-|x|}$.)

We will show that the following ensembles $\{A_n\}$ and $\{B_n\}$ meet the requirements of Proposition 4.36:

A_n**:** Sample x according to D_n. Sample z from $A(x)$. Output (x, z).

B_n**:** Sample x according to D_n. Sample z from $B(x)$. Output (x, z).

The statistical farness of these ensembles will follow from the farness of $A(x)$ and $B(x)$ on NO instances. The computational indistinguishability will follow from the computational indistinguishability of $A(x)$ and $B(x)$ on YES instances, together with the fact that it is hard to distinguish YES instances of Π from NO instances.

To formalize this intuition, we make some observations which follow from the fact that Π is hard-on-average (where here and throughout this proof, we write $\mathrm{neg}(n)$ to denote negligible functions):

1. $\Pr[D_n \notin \Pi_Y \cup \Pi_N] = \mathrm{neg}(n)$.
2. $\left|\Pr[D_n \in \Pi_Y] - \frac{1}{2}\right| = \mathrm{neg}(n)$ and $\left|\Pr[D_n \in \Pi_Y] - \frac{1}{2}\right| = \mathrm{neg}(n)$.
3. The ensembles $\{D_n^Y\}_{n\in\mathbb{N}}$ and $\{D_n^N\}_{n\in\mathbb{N}}$ obtained by conditioning D_n on being a YES or NO instance, respectively, are computationally indistinguishable.

Items 1 and 2 hold because otherwise the trivial algorithm that always outputs YES or the one that always outputs NO would decide Π correctly with nonnegligible advantage. Item 3 holds because a distinguisher between $\{D_n^Y\}$ and $\{D_n^N\}$ could be used to decide Π with nonnegligible advantage.

Claim 4.37 $\|A_n - B_n\| \geq 1/2 - \mathrm{neg}(n)$.

> **Proof of claim:** Since D_n must produce a NO instance of Π with probability at least $1/2 - \mathrm{neg}(n)$, $\|A_n - B_n\| \geq (1/2 - \mathrm{neg}(n)) \cdot (1 - 2^{-\Omega(n)}) = 1/2 - \mathrm{neg}(n)$.

Claim 4.38 *$\{A_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable.*

Proof of claim: Let M be any probabilistic polynomial-time algorithm. From the fact that $A(x)$ and $B(x)$ are computationally indistinguishable for YES instances, it follows that

$$|\Pr[M(x, A(x)) = 1 | x \in \Pi_Y] - \Pr[M(x, B(x)) = 1 | x \in \Pi_Y]| = \mathrm{neg}(n), \tag{3}$$

where these probabilities (and all those to follow) are taken over $x \leftarrow D_n$ and the coins of all algorithms (M, A, and B). By the computational indistinguishability of $\{D_n^Y\}$ and $\{D_n^N\}$, we also have

$$\begin{aligned} |\Pr[M(x, A(x)) = 1 | x \in \Pi_Y] - \Pr[M(x, A(x)) = 1 | x \in \Pi_N]| &= \mathrm{neg}(n) \\ |\Pr[M(x, B(x)) = 1 | x \in \Pi_Y] - \Pr[M(x, B(x)) = 1 | x \in \Pi_N]| &= \mathrm{neg}(n). \end{aligned}$$

Combining these with Equation 3, we see that all four conditional probabilities differ only by negligible amounts. Therefore,

$$\begin{aligned} &\Pr[M(x, A(x)) = 1] - \Pr[M(x, B(x)) = 1] \\ &\quad \leq |\Pr[M(x, A(x)) = 1 | x \in \Pi_Y] - \Pr[M(x, B(x)) = 1 | x \in \Pi_Y]| \\ &\quad\quad + |\Pr[M(x, A(x)) = 1 | x \in \Pi_N] - \Pr[M(x, B(x)) = 1 | x \in \Pi_N]| \\ &\quad\quad + 2\Pr[x \notin \Pi_Y \cup \Pi_N] \\ &\quad = \mathrm{neg}(n). \end{aligned}$$

This establishes the computational indistinguishability of $\{A_n\}$ and $\{B_n\}$.

Given these claims, the result now follows from Proposition 4.36. ■

5 Extensions to cheating-verifier zero knowledge

The focus of study in this paper has been the class of languages (or promise problems) possessing statistical zero-knowledge proofs *against an honest verifier*. However, in cryptographic applications, one usually wants the zero-knowledge condition to hold even against cheating verifier strategies that deviate arbitrarily from the specified protocol. There have been a number of results showing how to transform proof system which are statistical zero knowledge against the honest-verifier into ones that are statistical zero knowledge against cheating verifier strategies [BMO90, OVY93, Dam93, DGOW95, GSV98]. As advocated in [BMO90], one can use such transformations to translate results like ours about honest-verifier statistical zero knowledge to to the cheating-verifier definition. In this section, we discuss which of our results apply to the cheating-verifier class and the appropriate formulations in each case.

Of the transformations mentioned above, the result of [GSV98] is the only unconditional and unrestricted one; all the others use computational assumptions such as the existence of one-way functions or only apply to a restricted class of statistical zero-knowledge proofs. Since most of our results assert properties of the *class* SZK, much of their translation to the cheating-verifier class will immediately follow from [GSV98], since that transformation gives an equality between the honest-verifier and cheating-verifier classes. However, in order to translate results which assert the existence of honest-verifier *proof systems* with various properties, we must check that the transformation preserves those properties. Thus, in one instance, we will use transformation of [BMO90] instead, which will require making a complexity assumption.

Now, we give a formal definition of cheating-verifier statistical zero knowledge.

Definition 5.1 *An interactive protocol between a computationally unbounded prover P and a PPT verifier V is said to be a* (black-box) cheating-verifier statistical zero-knowledge *proof system for a promise problem Π if there exists a PPT simulator S and a negligible function α such that*

1. *If $x \in \Pi_Y$, then* $\Pr\left[(P,V)(x) = \mathtt{accept}\right] \geq 1 - c(|x|)$.
2. *If $x \in \Pi_N$, then for all P^*,* $\Pr\left[(P^*,V)(x) = \mathtt{accept}\right] \leq s(|x|)$.
3. *For all (even computationally unbounded) V^* and all $x \in \Pi_Y$, $\left\| S^{V^*}(x) - \mathrm{View}_{P,V^*}(x) \right\| \leq \alpha(|x|)$, where $S^{V^*}(x)$ denotes the output distribution of S with oracle access to V^*.*

As usual, $\alpha(\cdot)$ is called the simulator deviation*, $c(\cdot)$ the* completeness error*, and $s(\cdot)$ the* soundness error*.* cheating-ver SZK *denotes the class of promise problems possessing cheating-verifier statistical zero-knowledge proofs.*

The above definition is more stringent than the original definition in [GMR89] in several respects. The most important difference is that we require simulability for *all* verifier strategies, not just polynomial-time computable strategies. We also use a *black-box* notion of simulation, as introduced by [GO94]. That is, we say there should be a *single* simulator which works for all verifiers, given oracle access to that verifier, whereas the original definition in [GMR89] only asks that for every PPT verifier strategy, there exists a PPT simulator.[14] We also require that the simulator deviation is bounded by the same negligible function for all verifier strategies, instead of allowing a different negligible function for each verifier. Finally, we require that the simulator operate in strict polynomial time, whereas [GMR89] allows expected polynomial time. The main result of [GSV98] follows.

Theorem 5.2 ([GSV98]) cheating-ver SZK = SZK.

Theorem 5.2 is proven by transforming public-coin statistical zero-knowledge proofs against the honest verifier into public-coin statistical zero-knowledge proofs against cheating verifiers. By the private- to public-coin transformation of Okamoto (Theorem 3.10), this suffices to prove the theorem.

As one would expect, the conditional results of [BMO90], [OVY93], and [DGOW95, Part 2] do not meet our strong definition of cheating-verifier statistical zero knowledge. In the proof systems

[14] The notion of black-box zero knowledge is needed to make sense of a PPT machine simulating the behavior of a computationally unbounded verifier strategy.

that result from their transformations, the zero-knowledge condition only holds for PPT verifiers V^*, and the simulator deviation can depend on the verifier V^*. We will call a proof system meeting this weaker requirement an *cheating-PPT-verifier statistical zero-knowledge* proof system.

Now we examine which of our results are preserved under these two transformations.

The Completeness Theorem. Of course, since Theorem 5.2 gives an equality of classes, the Completeness Theorem extends to the cheating-verifier class:

Proposition 5.3 STATISTICAL DIFFERENCE *is complete for* cheating-ver SZK.

We now look at the applications of the Completeness Theorem, beginning with our results on efficient SZK proof systems in Corollary 4.2.

Simulator deviation and security parametrization. Both the transformations of [Oka00] and [GSV98] can be made to preserve a simulator deviation of $2^{-\Omega(n)}$. Applying these transformations to Corollary 4.2, we see that every language in SZK has a cheating-verifier statistical zero-knowledge proof with simulator deviation $2^{-\Omega(n)}$.

We can also consider a security-parametrized variant of cheating-verifier zero knowledge, analogous to the honest-verifier case (Definition 4.1): the protocol takes an extra parameter k (in unary) and the zero-knowledge condition demands that, for any verifier, the simulator deviation is less than $\alpha(k)$ for some negligible function α. The transformations of [Oka00, GSV98] both preserve the security-parametrization property, so we obtain:

Proposition 5.4 *Any promise problem in* SZK *has a cheating-verifier security-parametrized statistical zero-knowledge proof with simulator deviation* 2^{-k}.

Message complexity. Corollary 4.2 shows that every promise problem in SZK has a 2-message honest-verifier statistical zero-knowledge proof. Although the transformation of [GSV98] only multiplies the number of messages by a factor of two when applied to public-coin proof systems, the private- to public-coin transformation of [Oka00] increases the number of messages to polynomial even when applied to a constant-message protocol. However, if one is willing to make a computational assumption, then the transformation of [BMO90] applies and this transformation does preserve the message complexity up to a constant factor.

Proposition 5.5 *If the* DISCRETE LOGARITHM *problem is hard,*[15] *then every promise problem in* SZK *has a constant-message cheating-PPT-verifer statistical zero-knowledge proof system with soundness and completeness errors* 2^{-n}.

Proof: Let Π be any promise problem in SZK. From Corollary 4.2, we know that Π has a 2-message (honest-verifier) statistical zero-knowledge proof system. Repeating this protocol in parallel $O(n)$ times gives a constant-message proof system with soundness and completeness errors 2^{-n}. Note that parallel repetition preserves honest-verifier statistical zero knowledge. Now, apply the transformation of [BMO90], which yields a constant-message cheating-verifier statistical zero-knowledge proof system for Π, under the assumption that the discrete logarithm is hard. This transformation only increases the number of messages by a constant factor and preserves the completeness and soundness errors. ■

[15] See [BMO90] for a precise formulation of this assumption.

It is still open whether one can *unconditionally* prove that all of SZK has constant-message cheating-verifier proofs. We note that Goldreich and Krawcyzk [GK96] have shown some limitations on the message complexity of cheating-verifier zero-knowledge proofs (for problems outside BPP): If the proof system has *negligible soundness error* and is zero knowledge under black-box simulation, then it cannot consist of fewer than 4 messages. If, in addition, it is public coin, then it cannot consist of any constant number of messages.

Communication. Corollary 4.2 shows that every promise problem in SZK has a very communication-efficient honest-verifier statistical zero-knowledge proof, in that the prover only sends one bit to achieve completeness error $1 - 2^{-n}$ and soundness error $1/2 + 2^{-n}$. Unfortunately, none of the known transformations to cheating-verifier statistical zero knowledge preserve the amount of communication, so this result does not translate to cheating-verifier statistical zero knowledge.

Deterministic Prover. We note that the fact that the prover is deterministic in Corollary 4.2 *cannot* extend to cheating-ver SZK (unless SZK = BPP) [GO94].

Closure properties. Since Theorem 5.2 gives an equality of classes, any closure properties of the honest-verifier class (namely, Corollaries 4.3, 4.4, and 4.14, and Theorem 4.9) also hold for the cheating-verifier classs. So we immediately obtain the following:

Proposition 5.6 cheating-ver SZK *is closed under Karp reductions, complement, $\Phi(\cdot)$, and* NC^1 *truth-table reductions.*

Knowledge complexity. Cheating-verifier analogues of all the knowledge complexity classes discussed in Section 4.3 can be defined just as we have done for statistical zero knowledge. We adopt the same conventions as in Definition 5.1 — black-box strict polynomial-time simulation for all (not just PPT) verifier strategies, with the simulator deviation a negligible function independent of the verifier. We denote the cheating-verifier variant of a class C with cheating-ver C.

First, we show that honest-verifier and cheating-verifier statistical knowledge complexity in the *hint* sense coincide. To prove this, we observe one direction of the characterization of knowledge complexity in the hint sense given by Lemma 4.16 also holds for the cheating-verifier classes:

Lemma 5.7 *Let Π be any language and let $k(n)$ be any polynomially bounded function. Suppose there exists a promise problem $\Gamma \in$* cheating-ver SZK *(resp.,* cheating-ver PZK*) such that*

1. *$x \in \Pi_Y \Rightarrow$ there exists a such that $|a| = k(|x|)$ and $(x, a) \in \Gamma_Y$, and*
2. *$x \in \Pi_N L \Rightarrow$ for all a, $(x, a) \in \Gamma_N$.*

Then $\Pi \in$ cheating-ver $\mathsf{SKC}_{\mathsf{hint}}(k(n))$ *(resp.,* cheating-ver $\mathsf{PKC}_{\mathsf{hint}}(k(n))$*)*

The proof of Lemma 5.7 is the same as the corresponding direction of Lemma 4.16. The reason the other direction of Lemma 4.16 does not immediately apply to the cheating-verifier case is that the hint function may be different for each verifier. However, it will follow from the following:

Proposition 5.8 *For every polynomially-bounded function $k(n)$,*

$$\mathsf{SKC}_{\mathsf{hint}}(k(n)) = \text{cheating-ver } \mathsf{SKC}_{\mathsf{hint}}(k(n)).$$

Proof: Clearly, cheating-ver $\mathsf{SKC}_{\mathsf{hint}}(k(n)) \subset \mathsf{SKC}_{\mathsf{hint}}(k(n))$. Now suppose Π is any language in $\mathsf{SKC}_{\mathsf{hint}}(k(n))$, and let $\Gamma \in \mathsf{SZK}$ be the promise problem guaranteed by Lemma 4.16. Then, by Theorem 5.2, $\Gamma \in$ cheating-ver SZK. Applying Lemma 5.7, we see that $\Pi \in$ cheating-ver $\mathsf{SKC}_{\mathsf{hint}}(k(n))$. ■

Observe that we have actually proved something stronger: if $\Pi \in \mathsf{SKC}_{\mathsf{hint}}(k(n))$, then there is an proof system for Π with cheating-verifier statistical knowledge complexity $k(n)$ for which the *same hint function* can be used for every verifier. Also note that analogous results for the other variants of knowledge complexity do not appear to follow immediately from the fact that SZK = cheating-ver SZK.

Given Proposition 5.8, it follows immediately that Theorem 4.15 also holds for the cheating-verifier classes:

Proposition 5.9 *For any polynomially bounded function $k(n)$,*

$$\text{cheating-ver } \mathsf{SKC}_{\mathsf{hint}}(k(n) + \log n) = \text{cheating-ver } \mathsf{SKC}_{\mathsf{hint}}(k(n)).$$

In contrast, we do not know whether our results on the perfect knowledge complexity of SZK hold for the analogous cheating-verifier classes. To apply the same approach, one would have to analyze the (cheating-verifier) perfect knowledge complexity of the protocols obtained by performing the transformations of [Oka00] and [GSV98] on the protocol for SD. These transformations could conceivably increase the perfect knowledge complexity dramatically.

Hard-on-average problems and one-way functions. These results are stronger for the honest-verifier class, because the existence of a hard-on-average problem in the cheating-verifier class implies the existence of one in the cheating-verifier class (even without Theorem 5.2).

Acknowledgements

We are grateful to our advisor, Shafi Goldwasser, for getting us started on the topic of statistical zero knowledge and providing direction and advice throughout our work. We are indebted to Oded Goldreich for many enlightening conversations and subsequent collaboration on this topic, and his extensive help with the writing of this paper. We also thank Mihir Bellare, Tatsuaki Okamoto, Madhu Sudan, Luca Trevisan, Avi Wigderson, and anonymous reviewers for many helpful suggestions, clarifying discussions, and encouragement.

References

[ABV95] William Aiello, Mihir Bellare, and Ramarathnam Venkatesan. Knowledge on the average—perfect, statistical and logarithmic. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 469–478, Las Vegas, Nevada, 29 May–1 June 1995.

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, June 1991.

[AB84] Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 471–474, Washington, D.C., 1984.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[Bel97] Mihir Bellare. A note on negligible functions. Technical Report CS97-529, Department of Computer Science and Engineering, University of California at San Diego, March 1997. Also available from the Theory of Cryptography Library (`http://--------yu---ychh-yfy.sal-atore.rest/~tcryptol`).

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 482–493, Baltimore, Maryland, 14–16 May 1990.

[BP92] Mihir Bellare and Erez Petrank. Making zero-knowledge provers efficient. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 711–722, Victoria, British Columbia, Canada, 4–6 May 1992.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 3–5 1971 1971.

[DC96] Ivan Damgård and Ronald Cramer. On monotone function closure of perfect and statistical zero-knowledge. Theory of Cryptography Library: Record 96-03, 1996. `http://--------yu---ychh-yfy.sal-atore.rest/~tcryptol`.

[DGOW95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs. dishonest verifier in public coin zero-knowledge proofs. In *Proceedings of Crypto '95, Lecture Notes in Computer Science*, volume 403. Springer-Verlag, 1995.

[DGW94] Ivan Damgård, Oded Goldreich, and Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94–39, BRICS, November 1994. See Part 1 of [DGOW95].

[Dam93] Ivan B. Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 100–109. Springer-Verlag, 22–26 August 1993.

[DDPY94] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *35th Annual Symposium on Foundations of Computer Science*, pages 454–465, Santa Fe, New Mexico, 20–22 November 1994. IEEE.

[DDPY98] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image Density is complete for non-interactive-SZK. In *Automata, Languages and Programming, 25th International Colloquium*, Lecture Notes in Computer Science, pages 784–795, Aalborg, Denmark, 13–17 July 1998. Springer-Verlag. See also preliminary draft of full version, May 1999.

[DOY97] Giovanni Di Crescenzo, Tatsuaki Okamoto, and Moti Yung. Keeping the SZK-verifier honest unconditionally. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 17–21 August 1997.

[DSY00] Giovanni Di Crescenzo, Kouichi Sakurai, and Moti Yung. On zero-knowledge proofs: "from membership to decision". In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 255–264, Portland, OR, May 2000. ACM.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 327–343. JAC Press, Inc., 1989.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, 34(6):277–281, 28 May 1990.

[Gol95] Oded Goldreich. *Foundations of Cryptography (Fragments of a Book)*. Weizmann Institute of Science, February 1995. Available from `http://d-n-m-f-.sal-atore.restcc.uni-trier.de/eccc/`.

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 1–9, Dallas, 23–26 May 1998.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000. 30th Annual ACM Symposium on Theory of Computing (Dallas, TX, 1998).

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6:97–116, 1993.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR lemma. Technical Report TR95–050, Electronic Colloquium on Computational Complexity, March 1995. `http://d-n-m-f-.sal-atore.restcc.uni-trier.de/eccc`.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[GOP98] Oded Goldreich, Rafail Ostrovsky, and Erez Petrank. Computational complexity and knowledge complexity. *SIAM Journal on Computing*, 27(4):1116–1141, August 1998.

[GP99] Oded Goldreich and Erez Petrank. Quantifying knowledge complexity. *Computational Complexity*, 8(1):50–98, 1999.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 399–408, Dallas, 23–26 May 1998.

[GSV99] Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero-knowledge be made non-interactive?, or On the relationship of SZK and NISZK. In *Advances in Cryptology—CRYPTO '99*, Lecture Notes in Computer Science. Springer-Verlag, 1999, 15–19 August 1999. To appear.

[GV99] Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 54–73, Atlanta, GA, May 1999. IEEE Computer Society Press.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 73–90. JAC Press, Inc., 1989.

[GB00] Danny Gutfreund and Michael Ben-Or. Increasing the power of the dealer in non-interactive zero-knowledge proof systems. In *Advances in cryptology—ASIACRYPT '00 (Kyoto, Japan, 2000)*. Springer, Berlin, 2000. To appear.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396 (electronic), 1999.

[Hof95] Micha Hofri. *Analysis of Algorithms: Computational Methods & Mathematical Tools*. Oxford University Press, 1995.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In J. W. Thatcher and R. E. Miller, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, Inc., 1972.

[LLS75] Richard E. Ladner, Nancy A. Lynch, and Alan L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, December 1975.

[Lev73] Leonid A. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[Oka00] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *Journal of Computer and System Sciences*, 60(1):47–108, February 2000.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 133–138, Chicago, Illinois, 30 June–3 July 1991. IEEE Computer Society Press,.

[OVY93] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Proceedings of Eurocrypt '93, Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems*, 1993.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[PT96] Erez Petrank and Gábor Tardos. On the knowledge complexity of $\mathcal{NP}$. In *37th Annual Symposium on Foundations of Computer Science*, pages 494–503, Burlington, Vermont, 14–16 October 1996. IEEE.

[Sah00] Amit Sahai. *Frontiers in Zero Knowledge*. PhD thesis, Massachusetts Institute of Technology, September 2000.

[SV99] Amit Sahai and Salil Vadhan. Manipulating statistical difference. In Panos Pardalos, Sanguthevar Rajasekaran, and José Rolim, editors, *Randomization Methods in Algorithm Design (DIMACS Workshop, December 1997)*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 251–270. American Mathematical Society, 1999.

[SV97] Amit Sahai and Salil P. Vadhan. A complete promise problem for statistical zero-knowledge. In *38th Annual Symposium on Foundations of Computer Science*, pages 448–457, Miami Beach, Florida, 20–22 October 1997. IEEE.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992.

[Vad99] Salil P. Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, August 1999.

[Vad00] Salil P. Vadhan. On transformations of interactive proofs that preserve the prover's complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 200–207, Portland, OR, May 2000. ACM.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

A The Statistical Difference Metric

Proof of Fact 2.1: For any set $S \subset D$,

$$\begin{aligned} 2\left|\Pr[X \in S] - \Pr[Y \in S]\right| &= \left|\Pr[X \in S] - \Pr[Y \in S]\right| + \left|\Pr[X \notin S] - \Pr[Y \notin S]\right| \\ &= \left|\sum_{x \in S} (\Pr[X = x] - \Pr[Y = x])\right| + \left|\sum_{x \notin S} (\Pr[X = x] - \Pr[Y = x])\right| \\ &\leq \sum_{x \in S} \left|\Pr[X = x] - \Pr[Y = x]\right| + \sum_{x \notin S} \left|\Pr[X = x] - \Pr[Y = x]\right| \\ &= |X - Y|_1 . \end{aligned}$$

Equality is achieved by taking $S = \{x : \Pr[X = x] > \Pr[Y = x]\}$. ■

Proof of Fact 2.3:

$$\begin{aligned} \|(X_1, X_2) - (Y_1, Y_2)\| &\leq \|(X_1, X_2) - (Y_1, X_2)\| + \|(Y_1, X_2) - (Y_1, Y_2)\| \\ &= \frac{1}{2}|X_1 \otimes X_2 - Y_1 \otimes X_2|_1 + \frac{1}{2}|Y_1 \otimes X_2 - Y_1 \otimes Y_2|_1 \\ &= \frac{1}{2}|(X_1 - Y_1) \otimes X_2|_1 + \frac{1}{2}|Y_1 \otimes (X_2 - Y_2)|_1 \\ &= \frac{1}{2}|X_1 - Y_1|_1 \cdot |X_2|_1 + \frac{1}{2}|Y_1|_1 \cdot |X_2 - Y_2|_1 \\ &= \|X_1 - Y_1\| + \|X_2 - Y_2\| \end{aligned}$$

■

Proof of Fact 2.4: Let $A = (f, R)$ be any randomized procedure. Then, for any set $S \subset F$,

$$\begin{aligned} \left|\Pr[A(X) \in S] - \Pr[A(Y) \in S]\right| &= \left|\Pr[f(X \otimes R) \in S] - \Pr[f(Y \otimes R) \in S]\right| \\ &= \left|\Pr\left[X \otimes R \in f^{-1}(S)\right] - \Pr\left[Y \otimes R \in f^{-1}(S)\right]\right| \\ &\leq \|X \otimes R - Y \otimes R\| \\ &\leq \|X - Y\| + \|R - R\| \qquad \text{(by Fact 2.3)} \\ &= \|X - Y\|. \end{aligned}$$

Taking the maximum over all sets S completes the proof. ■

Proof of Fact 2.5: Let $T \subset D$ be the set of x's for which $\|X_2|_{X_1=x} - Y_2|_{Y_1=x}\| < \delta$. Now, let S be an arbitrary subset of $D \times E$ and, for every $x \in D$, define $S_x = \{y \in E : (x, y) \in S\}$. Then,

$$\begin{aligned} \Pr[X \in S] &\leq \Pr[X_1 \notin T] + \sum_{x \in T} \Pr[X_2 \in S_x | X_1 = x] \cdot \Pr[X_1 = x] \\ &< \epsilon + \sum_{x \in T} (\Pr[Y_2 \in S_x | Y_1 = x] + \delta) \cdot \Pr[Y_1 = x] \\ &\leq \epsilon + \delta + \Pr[Y \in S] . \end{aligned}$$

By symmetry, we also have $\Pr[Y \in S] < \epsilon + \delta + \Pr[X \in S]$. Since S was arbitrary, $\|X - Y\| < \epsilon + \delta$. ■

Proof of Fact 2.6: Let $S = \{x : (1 - \sqrt{\epsilon}) \Pr[X = x] \geq \Pr[Y = x]\}$, *i.e.* the set of x's for which the left-hand inequality in Fact 2.6 is violated. Then,

$$\begin{aligned} \Pr[Y \in S] &\leq \left(1 - \sqrt{\epsilon}\right) \Pr[X \in S] \\ &= \Pr[X \in S] - \sqrt{\epsilon} \cdot \Pr[X \in S]. \end{aligned}$$

Thus, $\sqrt{\epsilon} \cdot \Pr[X \in S] \leq \|X - Y\| < \epsilon$,so we must have $\Pr[X \in S] < \sqrt{\epsilon}$. A similar argument show that the right-hand inequality in Fact 2.6 is violated with probability less than $\sqrt{\epsilon}$. ■

B A Generic Complete Problem for PZK

In this section, we show how to obtain a complete promise problem for PZK directly from the definition of the class. However, in contrast to Statistical Difference, this problem will be essentially a restatement of the definition of the class and therefore of little use.

The complete promise problem for PZK is PZK-Generic, which we now define. An instance of PZK-Generic is a quadruple $(V, S, x, 1^t)$, where V is a description of an interactive probabilistic Turing machine and S is a description of a (noninteractive) probabilistic Turing machine. A yes instance is such a quadruple for which there exists a prover strategy P such that

1. The interaction between P and V on x takes at most t steps (including the computation time for V) and V accepts in this interaction.
2. The running time of S on input x is at most t.
3. S outputs `fail` with probability at most $1/2$, and conditioned on not failing, the output distribution of S is *identical* to V's view of the interaction with P on x.

A no instance is a quadruple such that for all prover strategies P,

1. The interaction between P and V on x takes at most t steps (including the computation time for V) and V rejects in this interaction.
2. The running time of S on input x is at most t.

Proposition B.1 PZK-Generic *is complete for* PZK.

Proof: First we show that every promise problem Π in PZK reduces to PZK-Generic. Let (P, V) be the perfect zero-knowledge proof system for Π with simulator S. Let $t(n)$ be a (polynomial) upper bound on both the running time of S and the number of steps of the interaction of P and V on inputs of length n. Then

$$x \mapsto (V, S, x, 1^{t(|x|)})$$

is a polynomial-time reduction from Π to PZK-Generic.

Now we argue that PZK-Generic $\in$ PZK. Consider the following descriptions of a verifier $\overline{V}$, a prover $\overline{P}$, and a simulator $\overline{S}$:

$\overline{V}(V, S, x, 1^t)$: When interacting with any machine, simulate V on input x.

$\overline{P}(V, S, x, 1^t)$**:** Exhaustively search for a prover strategy P for which V's view of $(P, V)(x)$ is identical to the output distribution of $S(x)$ (conditioned on $S(x) \neq \texttt{fail}$.) If one exists, follow that strategy, otherwise output `fail`.[16]

$\overline{S}(V, S, x, 1^t)$**:** Simulate S on input x.

It is easy to see that these definitions provide a perfect zero-knowledge proof system for PZK-GENERIC. ■

The problem with extending this example to SZK is Condition 3 for YES instances. "Identical" needs to be replaced by "negligible statistical difference," but it is not clear what negligible function to put there. We do not know how to get around this difficulty without using our Completeness Theorem, which implies that every problem in SZK has a statistical zero-knowledge proof with the *same* simulator deviation 2^{-n} (cf., Corollary 4.2).[17]

Another observation worth mentioning, pointed out to us by Bellare, Goldreich, and Sudan, is that PZK-GENERIC can be modified to obtain complete promise problems for cheating-ver PZK (as long as we restrict to "black-box" simulation) and also the various forms of PKC.

C An Example for GRAPH ISOMORPHISM

For illustrative purposes, here we explicitly describe what happens when the reduction to and proof system for STATISTICAL DIFFERENCE are applied to the well-known public-coin perfect zero-knowledge proof system for GRAPH ISOMORPHISM [GMW91]:

Perfect zero-knowledge proof system for GRAPH ISOMORPHISM**.**
Input: (G_0, G_1).

1. P sends V a random isomorphic copy H of G_0.
2. V picks $b \in \{0, 1\}$ at random and sends it to P.
3. P sends V a random isomorphism π between G_b and H, if one exists.
4. V checks that $\pi G_b = H$.

Simulator S, on input (G_0, G_1):

1. Pick random $b \in \{0, 1\}$ and a random permutation π.
2. Output $(\pi G_b, b, \pi)$.

Notice that the conversations output by S always make V accept.

If the reduction to SD from the proof of Lemma 3.8 is applied to the above protocol, the following distributions are obtained:

[16] Alternatively, $\overline{P}$ can act as the *simulation-based prover* (see Section 3.5).

[17] Note that the difficulty cannot be solved by the result of Bellare [Bel97], which states that any *countable* set of negligible functions is "dominated" by a single negligible function. The reason is that there are uncountably many problems in the promise-class SZK.

$A_0(G_0, G_1)$: Always output 1.
$B_0(G_0, G_1)$: Always output 1.
$A_1(G_0, G_1)$: Output $(\pi G_b, b)$ for a random permutation π and $b \in \{0,1\}$ chosen at random.
$B_1(G_0, G_1)$: Output $(\pi G_b, c)$ for a random permutation π and b and c chosen uniformly and independently from $\{0,1\}$.

Thus, $\|A_0(x) - B_0(x)\|$ always equals 0. $\|A_1(x) - B_1(x)\|$ is easily seen to be 0 if $G_0 \cong G_1$ and $1/2$ if $G_0 \not\cong G_1$. For the rest of this section, we ignore A_0 and B_0 since they are irrelevant.

If we now apply the protocol for SD from Section 3.3 to the distributions A_1 and B_1 (without first applying the Polarization Lemma), we obtain the following proof system (P', V') for GRAPH NONISOMORPHISM:

1. V' picks a random bit $d \in \{0,1\}$. If $d = 0$, V' chooses a random bit $b \in \{0,1\}$ and a random permutation π and sends $(\pi G_b, b)$ to P'. If $d = 1$, V' chooses random bits $b, c \in \{0,1\}$ and a random permutation π and sends $(\pi G_b, c)$ to P'.

2. P' receives message (H, b) from V'. P attempts to guess d as follows: If H is isomorphic to G_b, then P' guesses 0, else P' guesses 1.

3. V' accepts if the P' guesses d correctly.

Now, if G_0 is not isomorphic to G_1, then P' will guess correctly with probability $3/4$. However, if G_0 is isomorphic to G_1, then no prover can guess correctly with probability greater than $1/2$. The above protocol is of the same spirit as the standard GRAPH NONISOMORPHISM protocol [GMW91]. In both cases, the verifier randomly permutes one of the graphs to obtain a graph H and in order for the prover to succeed with probability greater than $1/2$, the prover needs to be able to tell which graph H came from.