On Transformations of Interactive Proofs that Preserve the Prover's Complexity

Salil Vadhan*

August 23, 2000

Abstract

Goldwasser and Sipser [GS89] proved that every interactive proof system can be transformed into a public-coin one (a.k.a., an Arthur–Merlin game). Their transformation has the drawback that the computational complexity of the prover's strategy is not preserved. We show that this is inherent, by proving that the same must be true of any transformation which only uses the original prover and verifier strategies as "black boxes." Our negative result holds even if the original proof system is restricted to be honest-verifier perfect zero knowledge and the transformation can also use the simulator as a black box.

We also examine a similar deficiency in a transformation of Fürer *et al.* [FGM+89] from interactive proofs to ones with perfect completeness. We argue that the increase in prover complexity incurred by their transformation is necessary, given that their construction is a black-box transformation which works regardless of the verifier's computational complexity.

Keywords: interactive proof systems, Arthur–Merlin games, zero-knowledge proofs, pseudorandom permutations

*MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. E-mail: `salil@theory.lcs.mit.edu`. URL: `http://theory.lcs.mit.edu/~salil`. Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. Much of this work was done when the author was visiting the Institute for Advanced Study and was supported by a DOD/NDSEG Graduate Fellowship.

1 Introduction

Since their introduction in 1985, the interactive proof systems of Goldwasser, Micali, and Rackoff [GMR89] and Arthur–Merlin games of Babai [BM88] have played a central role in complexity theory and cryptography. A surprising result of Goldwasser and Sipser [GS89] shows that these two important models are actually equivalent in expressive power. That is, every interactive proof system can be transformed into an Arthur–Merlin game (which is an interactive proof in which the verifier's messages consist solely of random coin flips). Although this result played an important role in subsequent theoretical work (cf., [FGM$^+$89, BHZ87, IY87, BGG$^+$88]), its applicability to cryptographic protocols is limited because the transformation does not preserve the complexity of the prover. In this paper, we demonstrate that this deficiency is inherent. We do this by showing that a wide class of transformations cannot transform general interactive proofs into public-coin ones (*i.e.,* Arthur-Merlin games) without increasing the prover's complexity.

Interactive Proofs. Informally, an *interactive proof* [GMR89] for a decision problem Π is an interactive protocol (P, V) by which a computationally unbounded *prover* P tries to convince a polynomial-time *verifier* V that some string x is a YES instance of Π.[1] Each of the two parties can privately flip coins, and exchange messages in polynomially many rounds of interactions, after which V either accepts or rejects. The definition requires:

1. (Completeness) If x is a YES instance, then V will accept with high probability after interacting with P.

2. (Soundness) If x is a NO instance, then V will reject with high probability after interacting with *any* strategy P^*.

A *public-coin* interactive proof system (or *Arthur–Merlin game*) [BM88] is one in which the verifer's messages at each round of interaction consist solely of random coin flips.

Intuitively, it seems that general interactive proofs should be much more powerful than public-coin ones. Indeed, several examples of interactive proofs, most notably the GRAPH NONISOMORPHISM proof system of [GMW91], appear to use private coins in an essential way. However, this intuition is incorrect, as Goldwasser and Sipser [GS89] demonstrated by giving a general method to transform any interactive proof into a public-coin one.

From both a theoretical and practical point of view, it is important to compare the complexity of the interactive proof systems produced by the Goldwasser–Sipser transformation with that of the original proof system. In some complexity measures, the transformation is very efficient. For example, it increases the round complexity of the proof system by only an additive constant. However, the transformation does not preserve the computational complexity of the prover strategy. For example, even if the original prover could be implemented in polynomial time given some auxiliary information (as is typically the case in cryptographic applications), the resulting prover is not guaranteed to have this property. We prove that this is inherent in the techniques used.

Black-box transformations. To obtain our negative result, we follow the approach pioneered by Impagliazzo and Rudich [IR89], and focus on a characteristic shared by most known transformations of interactive proof systems, including the one of Goldwasser and Sipser. Specifically, these transformations only use the original prover and verifier strategies as "black boxes". That is, the new protocol only exploits the

[1] We allow Π to be a promise problem [ESY84], rather than just a language. Formally, a *promise problem* $\Pi = (\Pi_Y, \Pi_N)$ is a pair of disjoint sets of strings, referred to as YES and NO instances, respectively.

input-output behavior of these strategies, rather than the particular algorithms used to compute them. More precisely, if (P, V) is the original interactive proof and (P', V') is the new interactive proof, then

1. The strategy of V' on input x can be computed by a polynomial-time algorithm given oracle access to the strategy of V on input x.

2. The strategy of P' on input x can be computed by a (not necessarily efficient) algorithm given oracle access to the strategies of P and V on input x.

Here, by the *strategy* of a party $A \in \{P, V\}$ on input x, we mean the function A_x which takes A's random coins and the history of messages exchanged and outputs A's next message. We call a transformation $(P, V) \mapsto (P', V')$ satisfying the above two properties a *black-box transformation*. Note that the algorithms used to compute P' and V' do not even explicitly look at the input x; the role played by the input is limited to its effect on the strategies of P and V (i.e., the input to P' and V' is just the length of x in unary). We say that a black-box transformation *preserves the prover's complexity* if the strategy of P' on input x can in fact be computed in polynomial time given oracle access to the strategies P and V on input x.

With these definitions, we can state our main result:

Theorem 1 *If one-way functions exist, then there is no black-box transformation from private-coin interactive proofs to public-coin ones that preserves the prover's complexity.*

A natural question is whether "current techniques" are actually limited to black-box transformations. First, we note that, in addition to the Goldwasser–Sipser transformation, most other general tranformations of interactive proofs are also black-box transformations. Examples include the Collapse Theorem of Babai and Moran [BM88], the transformation of [FGM+89] from interactive proofs to ones with perfect completeness, and transformations of honest-verifier zero-knowledge proofs to general zero-knowledge proofs [BMO90, OVY93, DGOW95, GSV98]. The only exceptions we know of are those that exploit complete problems, such as [GMW91, BGKW88, LFKN92, Sha92], and typically this approach increases complexity "to the maximum." For example, another way to prove that every problem possessing an interactive proof also has a public-coin interactive proof would be to combine the inclusion $\mathbf{IP} \subset \mathbf{PSPACE}$ with the direct public-coin interactive proof for the $\mathbf{PSPACE}$-complete problem QUANTIFIED BOOLEAN FORMULA [LFKN92, Sha92]. This approach necessarily yields interactive proofs whose complexity is that of QUANTIFIED BOOLEAN FORMULA, regardless of the complexity of the original interactive proof.

Zero-knowledge proofs. The cryptographic interest in interactive proofs focuses primarily on *zero-knowledge proofs* [GMR89], which can be informally described as interactive proofs in which the verifier learns nothing from the interaction other than the fact that the assertion being proven is true. This property is formalized by requiring that there is an efficient algorithm, called a *simulator*, whose output distribution (on YES instances) is "similar" to the verifier's view of the interaction. Intuitively, this means that the verifier learns nothing since it could run the simulator instead of interacting with the prover.

There are several choices in the definition of zero-knowledge proofs that give rise to notions of varying strength. Regarding the quality of simulation, there are three common interpretations of "similarity" for probability distributions, which lead to the notions of *perfect* zero knowledge, *statistical* zero knowledge, and *computatational* zero knowledge. Another choice is whether we should only require that the verifier learns nothing if it follows the specified protocol, or whether we should demand the same for cheating verifier strategies that can deviate arbitrarily from the specified protocol. The former is known as *honest-verifier zero knowledge*, whereas the latter is often called *general zero knowledge*.

The Goldwasser–Sipser transformation does not preserve any sort of zero knowledge property. This was remedied by Okamoto [Oka96], who showed how to transform "honest-verifier statistical zero-knowledge proofs" into ones which use only public coins. Okamoto's transformation provided a crucial starting point for a number of subsequent works on statistical zero knowledge (cf., [SV97, GSV98, Vad99]). Like the Goldwasser–Sipser transformation, neither Okamoto's transformation nor its later simplifications [GV99, Vad99] preserve the complexity of the prover.

The statement of Theorem 1 does not immediately apply to the zero knowledge setting, because the transformations of zero-knowledge proofs mentioned above use black-box access to the simulator in addition to the prover and verifier. However, our proof gives something much stronger than what is stated in Theorem 1, and does imply an analogous result for transforming zero-knowledge proofs.

Theorem 2 (Thm. 1, strengthened) *Assume one-way functions exist. Then there is no black-box transformation from honest-verifier perfect zero-knowledge proof systems to public-coin proof systems that preserves the prover's complexity, even if the new prover and verifier are also allowed black-box access to the simulator for the original proof system.*

In fact, we exhibit a specific problem and honest-verifier perfect zero-knowledge proof on which any such transformation must fail. The problem is called DISJOINT SUPPORT, and is a restriction of STATISTICAL DIFFERENCE, the complete problem for statistical zero knowledge given in [SV97].

Unconditional results. The assumption that one-way functions exist can be removed from both Theorems 1 and 2 if we augment the definition of black-box transformation with another property satisfied by all the black-box transformations we have mentioned. Namely, the transformations in [GS89, Oka96, GV99, Vad99] all work even when the original verifier is not polynomial time. Clearly, in such a situation we cannot hope for the new verifier to run in polynomial time. But it is still meaningful to require that the new verifier runs in polynomial time when given oracle access to the original verifier's strategy on a given input, and that completeness and soundness are preserved on an input-by-input basis. We call black-box transformation that satisfies this property a *strong black-box transformation*. We can prove analogues of Theorems 1 and 2 for strong black-box transformations without any computational assumption.

Perfect completeness. An interactive proof system is said to have *perfect completeness* if the verifier accepts with probability 1 when interacting with the prover on YES instances. Fürer *et al.* [FGM+89] showed that every interactive proof can be transformed into one with perfect completeness. Their transformation does not preserve the prover's complexity. We show that this is inherent in the fact that their construction is a strong black-box transformation.

Proposition 3 *There is no strong black-box transformation of general interactive proofs into ones with perfect completeness that preserves that prover's complexity.*

The restriction to *strong* black-box transformations is important in our proof of Proposition 3. In fact, recent results on derandomization give (non-strong) black-box transformations in cases where we have ruled out strong black-box transformations. We discuss this connection with derandomization in more detail in Section 5.

Additional Related Work. Kilian [Kil90] introduced the terminology "robust transformations" for transformations of interactive proof systems that preserve the complexity of the prover, and gave a first step towards achieving a robust transformation from interactive proofs to zero-knowledge proofs. The definition of a robust transformation does not require that the original verifier strategy be accessed only as a black box. The complexity of the prover in interactive proofs was previously studied by Bellare and Goldwasser [BG94]. They showed that, under a complexity-theoretic assumption, there is a problem Π in $\mathbf{NP}$

which is harder to prove than it is to decide; that is, no interactive proof for Π has a prover which can be implemented in polynomial time given an oracle for deciding Π.

2 The main result

Our proof of Theorem 2 is based on analyzing the effect of a black-box transformation on a proof system for a problem called DISJOINT SUPPORT. The definition of DISJOINT SUPPORT involves probability distributions encoded by circuits which sample from them. More precisely, if X is a Boolean circuit with m input gates and n output gates, *the probability distribution encoded by* X is the distribution on $\{0,1\}^n$ induced by feeding X the uniform distribution on $\{0,1\}^m$ and taking the output. For notational convenience, we also denote this probability distribution by X. We write $X \equiv Y$ to indicate that the probability distributions encoded by circuits X and Y are identical. The *support* of probability distribution X on a universe $\mathcal{U}$ is the set $\mathrm{Supp}(X) \subset \mathcal{U}$ of points which are assigned nonzero probability mass under X.

Definition 4 DISJOINT SUPPORT *(*DS*) is the promise problem* $\mathrm{DS} = (\mathrm{DS}_Y, \mathrm{DS}_N)$ *given by:*

$$\begin{aligned} \mathrm{DS}_Y &= \{(X,Y) \ : \ \mathrm{Supp}(X) \cap \mathrm{Supp}(Y) = \emptyset\} \\ \mathrm{DS}_N &= \{(X,Y) \ : \ X \equiv Y\} \end{aligned}$$

In this definition, both X and Y are circuits encoding probability distributions in the manner described above.

DISJOINT SUPPORT is a restriction of STATISTICAL DIFFERENCE (SD), the complete problem for statistical zero knowledge given in [SV97]. The interactive proof system for DS we consider is given in Protocol 5. It is a restriction of the proof system for SD, which in turn is based on ideas from the proof systems for QUADRATIC NONRESIDUOSITY [GMR89] and GRAPH NONISOMORPHISM [GMW91].

Protocol 5: Proof system (P, V) for DISJOINT SUPPORT

Input: Circuits X_0 and X_1 (each with m input gates and n output gates)

1. V: Select $b \leftarrow \{0,1\}$. Obtain a sample $x \leftarrow X_b$ (by choosing $r \leftarrow \{0,1\}^m$ and letting $x = X_b(r)$). Send x to P.
2. P: If $x \in \mathrm{Supp}(X_1)$, let $c = 1$. Else let $c = 0$. Send c to V.
3. V: If $c = b$, accept. Otherwise, reject.

In order to understand what it means to apply a black box transformation to this protocol, we must determine what power oracle access to the verifier strategy and prover strategy gives. For a fixed input (X_0, X_1), the verifier strategy has two components:

1. A function that takes the verifier's random coins (b, r) and outputs $X_b(r)$.
2. A function that takes the verifier's random coins (b, r) and a prover message c, and outputs `accept` or `reject` according to whether $b = c$.

Clearly, the second function provides no power as an oracle, since it is just an equality test. Having oracle access to the first function is equivalent to having oracle access to each circuit X_0 and X_1 individually. The prover strategy is simply a membership oracle for $\mathrm{Supp}(X_1)$; that is the oracle returns 1 on input x iff $x \in \mathrm{Supp}(X_1)$.

Motivation. Suppose this proof system could be converted to a public-coin one via a black-box transformation. This means that there are polynomial-time algorithms M and A, such that $(M^{X_0,X_1,\mathrm{Supp}(X_1)}, A^{X_0,X_1})$ gives a public-coin proof system for DISJOINT SUPPORT. Recall that M and A are not given X_0 and X_1 explicitly as input, though they may run in time polynomial in the size of the input $|(X_0, X_1)|$.

For intuition as to why such an M and A cannot exist, let us suppose that X_0 and X_1 can be arbitrary one-to-one mappings from $\{0,1\}^k$ to $\{0,1\}^{3k}$ (such that either $X_0 \equiv X_1$ or $\mathrm{Supp}(X_0) \cap \mathrm{Supp}(X_1) = \emptyset$) and that M and A are only given running time polynomial in k. In other words, we are no longer requiring that X_0 and X_1 are given by small circuits. In fact, let us suppose that X_0 and X_1 are selected uniformly at random among all mappings satisfying the stated conditions.

First, we argue that if M never queries the oracle for $\mathrm{Supp}(X_1)$, then we are done. If M never queries the oracle for $\mathrm{Supp}(X_1)$, then the interaction between M and A can be simulated by a probabilistic polynomial-time algorithm B just given oracle access to X_0 and X_1. By completeness and soundness, such a B can determine whether $X_0 \equiv X_1$ or $\mathrm{Supp}(X_0) \cap \mathrm{Supp}(X_1) = \emptyset$, just given oracle access to X_0 and X_1. We claim this is impossible. This is because B's view will be statistically independent of whether it is given a YES or NO instance; in both cases, B will simply see distinct, (almost) uniformly distributed elements of $\{0,1\}^{3k}$ at each point it queries X_0 and X_1.

Therefore, it suffices to show that M's oracle access to $\mathrm{Supp}(X_1)$ is "useless" in the sense that we can remove M's queries to $\mathrm{Supp}(X_1)$ without affecting the completeness or soundness of the proof system. Here is where we exploit the fact that the proof system is public coin. Consider the first query x that M makes to $\mathrm{Supp}(X_1)$. If M has previously obtained x by evaluating X_1 at some point, then the response of the oracle will certainly be 1, so it need not ask the query. We claim that, with high probability over the choice of a random YES instance, M cannot generate any other queries that lie in $\mathrm{Supp}(X_1)$. Intuitively, this is because the points at which it has queried X_1 give M essentially no information about other points in $\mathrm{Supp}(X_1)$, and X_0 is essentially independent of X_1 (since two independently selected mappings from $\{0,1\}^n \to \{0,1\}^{3n}$ will have disjoint ranges with high probability). Note that A does not provide M with any assistance in generating a useful query, since A only sends M random coin flips. We conclude that we can remove M's queries to $\mathrm{Supp}(X_1)$ only slightly reducing the probability that A accepts on a random YES instance. So, completeness is preserved on almost all YES instances, and soundness is preserved since we have not modified A. This yields a contradiction.

The main lemma. While for motivation above, we allowed X_0 and X_1 to have much higher complexity than the algorithms M and A, we want to prove that a black-box transformation must fail even if M and A are given running time polynomial in the circuit sizes of X_0 and X_1. We will show how to construct efficient, pseudorandom versions of the mappings X_0 and X_1 used above, and these will suffice to complete the proof. The following lemma states the properties that are needed to prove our main theorem.

Lemma 6 *If one-way functions exist, then there are ensembles of distributions $\{\mathcal{D}_Y^k\}_{k\in\mathbb{N}}$ and $\{\mathcal{D}_N^k\}_{n\in\mathbb{N}}$ on pairs of circuits such that:*

1. *$\mathcal{D}_Y^k$ and $\mathcal{D}_N^k$ only produce pairs (X_0, X_1) such that X_0 and X_1 both map $\{0,1\}^k$ to $\{0,1\}^{3k}$ and both are circuits of size at most* $\mathrm{poly}(k)$.
2. $\Pr\left[\mathcal{D}_N^k \in \mathrm{DS}_N\right] = 1$ *and* $\Pr\left[\mathcal{D}_Y^k \in \mathrm{DS}_Y\right] \geq 1 - 2^{-k}$.

3. *For every probabilistic polynomial-time algorithm B, there is a negligible*[2] *function α such that*

$$\left| \Pr_{(X_0,X_1)\leftarrow \mathcal{D}_Y^k}\left[B^{X_0,X_1}(1^k)=1\right] - \Pr_{(X_0,X_1)\leftarrow \mathcal{D}_N^k}\left[B^{X_0,X_1}(1^k)=1\right]\right| \leq \alpha(k).$$

4. *For every probabilistic polynomial-time algorithm M, the probability that M succeeds in the following experiment is bounded by negligible function of k:*
   (a) *Select $(X_0,X_1)\leftarrow \mathcal{D}_Y^k$.*
   (b) *Run $M^{X_0,X_1}(1^k)$ to obtain output x.*
   (c) *M succeeds if $x \in \mathrm{Supp}(X_1)$ and M did not obtain x as a response to a query to the X_1-oracle.*

We defer the proof of this lemma to the next section and proceed with the proof of Theorem 1. The proof essentially follows the motivation given above.

Proof of Theorem 1: Suppose the theorem is false. Then there are polynomial-time algorithms M and A such that $(M^{X_0,X_1,\mathrm{Supp}(X_1)}, A^{X_0,X_1})$ gives a public-coin proof system for DISJOINT SUPPORT. Recall that M and A are not given X_0 and X_1 explicitly as input, though they may run in time polynomial in the size of the input $|(X_0,X_1)|$.

We may assume that the completeness and soundness errors are at most $1/3$. In particular, the probability that A accepts in the following experiment is at least $2/3$:

Experiment I

1. Select $(X_0,X_1)\leftarrow \mathcal{D}_Y^k$.
2. Run the interactive protocol between $M^{X_0,X_1,\mathrm{Supp}(X_1)}(1^k)$ and $A^{X_0,X_1}(1^k)$, at the end of which A accepts or rejects.

We will show that there is a probabilistic polynomial-time algorithm M_2 such that if $M^{X_0,X_1,\mathrm{Supp}(X_1)}$ is replaced by $M_2^{X_0,X_1}$ in the above experiment, A will still accept with probability at least $2/3 - \mathrm{neg}(k)$. M_2 is defined as follows: $M_2^{X_0,X_1}$ simply simulates $M^{X_0,X_1,\mathrm{Supp}(X_1)}$ until M tries to ask a query x to the $\mathrm{Supp}(X_1)$-oracle. If x was previously obtained as a response to a query to the X_1-oracle, M_2 feeds M the response 1, otherwise it gives the response 0.

To show that A still accepts with probability $2/3 - \mathrm{neg}(k)$ when M_2 is used, it suffices to show that M_2 answers all of M's queries correctly with all but negligible probability. If this weren't the case, then the following algorithm M_3 would violate Property 4 in Lemma 6: $M_3^{X_0,X_1}$ first chooses i uniformly from $\{1,\ldots,q(k)\}$, where $q(k)$ is polynomial bound on the number of queries M makes to the $\mathrm{Supp}(X_1)$-oracle. Then M_3 proceeds exactly as M_2 until M makes its i'th query x to the $\mathrm{Supp}(X_1)$-oracle, at which point M_3 halts and outputs x. Now, whenever M_2 would answer some query incorrectly, M_3 succeeds if it chooses i corresponding to the first incorrect response, which happens with probability $1/q(k)$.

Therefore, replacing $M^{X_0,X_1,\mathrm{Supp}(X_1)}$ with $M_2^{X_0,X_1}$ will decrease A's acceptance probability in Experiment I by at most a negligible amount. Now consider the probabilistic polynomial-time algorithm B which, when given oracle access to X_0 and X_1, simulates the interaction between $M_2^{X_0,X_1}$ and A^{X_0,X_1} and outputs 1 iff A accepts. When given $(X_0,X_1)\leftarrow \mathrm{DS}_Y$, B will output 1 with probability at least $2/3 - \mathrm{neg}(k)$ by what we have just shown. When given $(X_0,X_1)\leftarrow \mathrm{DS}_N$, B will output 1 with probability at most $1/3$, by the soundness of A^{X_0,X_1}. This contradicts Property 3 of Lemma 6. ■

[2]A function $\alpha : \mathbb{N}\to[0,1]$ is *negligible* if for every polynomial $p:\mathbb{N}\to\mathbb{N}$, $\alpha(n) < 1/p(n)$ for sufficiently large n.

The above proof highlights the difficulty faced by a prover in a public-coin proof system. It is more difficult for the prover to make "useful" queries to an oracle, since it must essentially generate the queries on its own; the verifier only provides random coin flips.

3 Proof of Lemma 6

The construction of circuits we need is based on pseudorandom permutations [LR88].

Definition 7 *Let $\mathcal{P} = \bigcup_k \mathcal{P}_k$, where $\mathcal{P}_k$ is a set of permutations $\pi_s : \{0,1\}^k \to \{0,1\}^k$ indexed by seeds $s \in \{0,1\}^{k^c}$ (for some constant $c > 0$). $\mathcal{P}$ is said to be a family of* strong pseudorandom permutations *if the following properties hold:*

1. *Given $s \in \{0,1\}^{k^c}$ and $x \in \{0,1\}^k$, $\pi_s(x)$ and $\pi_s^{-1}(x)$ can be evaluated in time $\mathrm{poly}(k)$.*
2. *For every probabilistic polynomial-time algorithm A, there is a negligible function α such that*

$$\left| \Pr_{s \leftarrow \{0,1\}^{k^c}} \left[A^{\pi_s, \pi_s^{-1}}(1^k) = 1 \right] - \Pr_{\pi \leftarrow \mathcal{G}_{k,k}} \left[A^{\pi, \pi^{-1}}(1^k) = 1 \right] \right| \leq \alpha(k),$$

   where $\mathcal{G}_{k,k}$ denotes the set of all permutations on $\{0,1\}^k$.

Luby and Rackoff [LR88] showed how to construct a pseudorandom permutation family based on any pseudorandom function family, which in turn can be constructed from any one-way function [GGM86, HILL99]. Simplified constructions and analyses of pseudorandom permutations can be found in [NR97].

Theorem 8 ([LR88, GGM86, HILL99]) *If there exist one-way functions, then there exist pseudorandom permutation families.*

From pseudorandom permutation families, it is easy to construct pseudorandom one-to-one functions.

Lemma 9 *Assume one-way functions exist. Then there is a family of functions $\mathcal{F} = \bigcup_k \mathcal{F}_k$, where $\mathcal{F}_k$ is a set of one-to-one functions $f_s : \{0,1\}^k \to \{0,1\}^{3k}$ indexed by seeds $s \in \{0,1\}^{k^c}$ (for some constant $c > 0$), with the following properties:*

1. *Given s and $x \in \{0,1\}^k$, $f_s(x)$ can be evaluated in time $\mathrm{poly}(k)$.*
2. *For every $x \in \{0,1\}^k$, $f_s(x)$ is distributed uniformly in $\{0,1\}^{3k}$ (over the choice $s \leftarrow \{0,1\}^{k^c}$).*
3. *For every probabilistic polynomial-time algorithm A, there is a negligible function α such that*

$$\left| \Pr_{s \leftarrow \{0,1\}^{k^c}} \left[A^{f_s}(1^k) = 1 \right] - \Pr_{f \leftarrow \mathcal{G}_{k,3k}} \left[A^{f}(1^k) = 1 \right] \right| \leq \alpha(k),$$

   where $\mathcal{G}_{k,3k}$ denotes the set of all one-to-one functions from $\{0,1\}^k$ to $\{0,1\}^{3k}$.

4. *For every probabilistic polynomial-time algorithm A, the probability that A succeeds in the following experiment is bounded by a negligible function of k:*
   (a) *Choose $s \leftarrow \{0,1\}^{k^c}$.*
   (b) *Execute $A^{f_s}(1^k)$ to obtain output x.*

(c) A succeeds *if x is in the range of f_s and A did not obtain x as a response to a query to the f_s-oracle.*

Proof: By Theorem 8, there is a strong pseudorandom permutation family $\mathcal{P} = \bigcup \mathcal{P}_k$. We obtain our function family $\mathcal{F} = \bigcup \mathcal{F}_k$ as follows: A function in $f_{s,y} \in \mathcal{F}_k$ is indexed by a permutation $\pi_s \in \mathcal{P}_{3k}$ and a string $y \in \{0,1\}^{3k}$, and is defined by $f_{s,y}(x) = \pi_s(x \circ 0^{2k}) \oplus y$. $f_{s,y}$ is one-to-one because π is a permutation. $f_{s,y}(x)$ varies uniformly over $\{0,1\}^{3k}$ just over the choice of y, so Property 2 holds. Properties 1 and 3 are straightforward to verify from the properties of pseudorandom permutations. Property 4 also follows from the definition of strong pseudorandom permutations: It is easy to see that A would succeed with negligible probability if f were constructed using a truly random permutation π instead of the pseudorandom permutation π_s, even if A is also given the random translation y. Since the success of A can be checked using oracle access to π^{-1} (together with y), it can be used to build a distinguisher for the strong pseudorandom permutation family $\mathcal{P}$. ■

We now prove Lemma 6.

Proof of Lemma 6: Let $\mathcal{P} = \bigcup \mathcal{P}_k$ be a family of strong pseudorandom permutations and let $\mathcal{F} = \bigcup \mathcal{F}_k$ be the family of functions guaranteed by Lemma 9. The distributions $\mathcal{D}_Y^k$ and $\mathcal{D}_N^k$ are defined as follows:

$\mathcal{D}_Y^k$: Select f_0 and f_1 independently from $\mathcal{F}_k$. Let X_0 and X_1 be the circuits evaluating these functions. Output (X_0, X_1).

$\mathcal{D}_N^k$: Select f randomly from $\mathcal{F}_k$ and π randomly from $\mathcal{P}$. Let X_0 be the circuit evaluating f, and let X_1 be the circuit evaluating $f \circ \pi$. Output (X_0, X_1).

We now prove that all the properties required by Lemma 6 hold. The circuit sizes of X_0 and X_1 are bounded by $\mathrm{poly}(k)$ by the efficiency of $\mathcal{P}$ and $\mathcal{F}$, so Property 1 holds. For Property 2, note that f and $f \circ \pi$ both induce the uniform distribution on the range of f, so $\mathcal{D}_N^k$ always produces NO instances. To see that $\mathcal{D}_Y^k$ almost always produces YES instances, note that for every $x, y \in \{0,1\}^k$, the probability that $f_0(x) = f_1(y)$ is 2^{-3k} (by Lemma 9, Item 2). Hence, the probability that the range of f_0 and f_1 intersect is at most $(2^k)^2 \cdot 2^{-3k} = 2^{-k}$ and Property 2 holds. For the indistinguishability of $\mathcal{D}_Y^k$ and $\mathcal{D}_N^k$ (Property 3), observe that a polynomial-time algorithm would have only exponentially small advantage in distinguishing the two distributions if all the functions used (f_0, f_1, f, and π) to construct X_0 and X_1 were truly random one-to-one functions. The indistinguishability then follows from the pseudorandomness of the families $\mathcal{P}$ and $\mathcal{F}$. Finally, Property 4 follows immediately from Lemma 9, Item 4. ■

4 Extensions

As mentioned in the introduction, the proof system for DISJOINT SUPPORT on which our construction is based is actually honest-verifier perfect zero knowledge. By definition, this means that there is a probabilistic polynomial-time simulator which, when fed a YES instance x, produces an output distribution which is *identical* to the verifier's view of the interaction with the prover on input x. (The verifier's *view* is a random variable $(\tau; r)$ consisting of a transcript τ of all the messages exchanged together with the verifier's random coins r.) Such a simulator for Protocol 5 is given by Algorithm 10.

Algorithm 10: Simulator for DISJOINT SUPPORT **proof system**

Input: Circuits X_0 and X_1 (each with m input gates and n output gates)

1. Select $b \leftarrow \{0,1\}$. Choose $r \leftarrow \{0,1\}^m$ and let $x = X_b(r)$.
2. Let $c = b$.
3. Output $(x, c; b, r)$

It is immediate to verify that, on YES instances of DS, this simulator's output distribution is identical to the V's view of Protocol 5. Now, a transformation from honest-verifier perfect zero-knowledge proofs to public-coin proofs, such as the one given by Okamoto [Oka96], might also make use of black-box access to the simulator. But, for this proof system, black-box access to the simulator is equivalent to black-box access to the verifier. For a fixed input (X_0, X_1), the simulator simply takes a pair (r, b) and outputs $(X_b(r), b; b, r)$. As was the case with the verifier, having oracle access to this function is equivalent to having oracle access to X_0 and X_1 individually. Therefore, having oracle access to the simulator does not help in giving a black-box transformation. This establishes Theorem 2.

A second observation about our construction is that the assumption that one-way functions exist is unnecessary if we only want to rule out *strong* black-box transformations. Recall that a strong black-box transformation of interactive proofs is one that works regardless of the computational complexity of the verifier's strategy; the Goldwasser–Sipser transformation [GS89] is an example of such a transformation. In such a case, the input x is irrelevant, except that it bounds the number of random coins used by the two parties and the total amount of communication between them to be a polynomial in $|x|$. To show that there do not exist strong black-box transformations from private coins to public coins that preserve the prover's complexity, it suffices to have an analogue of Lemma 6 in which the condition on the sizes of the circuits X_0 and X_1 is removed (though their input and output lengths should remain k and $3k$, respectively). Such an analogue can be proven *unconditionally* using truly random permutations and one-to-one functions rather than pseudorandom ones.

5 Perfect completeness

Recall that an interactive proof is said to have *perfect completeness* if the verifier accepts with probability 1 on YES instances. In this section, we discuss black-box transformations from interactive proofs to ones with perfect completeness. Fürer, Goldreich, Mansour, Sipser, and Zachos [FGM+89] have given such a transformation, in fact a strong black-box transformation, but it does not preserve the prover's complexity. Below, we explain why there can be no strong black-box transformation that preserves the prover's complexity (Proposition 3).

Lemma 11 *Suppose there is a black-box transformation from interactive proofs to ones with perfect completeness that preserves the prover's complexity. Then, for any interactive proof system (P, V) for any problem Π, there is a probabilistic polynomial-time algorithm A such that:*

1. $x \in \Pi_Y \Rightarrow \Pr\left[A^{P_x, V_x}(1^{|x|})\right] = 1.$
2. $x \in \Pi_N \Rightarrow \Pr\left[A^{P_x, V_x}(1^{|x|})\right] \leq 1/2.$

Above, P_x and V_x denote the strategies of P and V on input x. If the transformation is a strong *black-box transformation, then the same conditions hold even if V is allowed to be computationally unbounded.*

Proof: A simulates the transformed proof system (P', V') on input x and outputs 1 if V' accepts. ■

A subclass of interactive proofs are ones in which the verifier never interacts with the prover — these are equivalent to $\mathbf{BPP}$ algorithms. In such a case P_x is useless, so Lemma 11 says that A^{V_x} can decide Π with one-sided error. Since V is polynomial time, this implies that $\Pi \in \mathbf{co\text{-}RP}$. Thus, we conclude that $\mathbf{BPP} \subset \mathbf{co\text{-}RP}$. Since $\mathbf{BPP}$ is closed under complement, we have:

Proposition 12 *If there is a black-box transformation from interactive proofs to ones with perfect completeness that preserves the prover's complexity, then* $\mathbf{BPP} = \mathbf{ZPP}$.

What does this reasoning give for *strong* black-box transformations? In this case, the verifier strategy V_x can be an arbitrary function from the space of its random coin tosses (say $\{0,1\}^m$) to $\{\mathtt{accept}, \mathtt{reject}\}$ which either accepts at least 2/3 of its inputs or rejects at least 2/3 of its inputs. Lemma 11 says that a probabilistic $\mathrm{poly}(m)$-time algorithm can distinguish between these two cases with one-sided error, given only oracle access to V_x. This is impossible if there are no restrictions placed on V_x. This proves Proposition 3.

This provides an explanation for why the transformation of Fürer *et al.* does not preserve the prover's complexity, but we used the restriction to strong black-box transformations in an essential way. The conclusion for non-strong black-box transformations ($\mathbf{BPP} = \mathbf{ZPP}$) was much weaker, in fact quite plausible. This is not an accident. Under plausible intractability assumptions, a series of works (beginning with [NW94] and culminating in [IW97]) have constructed pseudorandom generators $G : \{0,1\}^{O(\log m)} \to \{0,1\}^m$ whose output looks pseudorandom to any algorithm running in time, say, m^2. Such a pseudorandom generator can be used to give a black-box derandomization of any $\mathbf{BPP}$ algorithm by running the algorithm on all possible outputs of the generator. The resulting algorithm is deterministic, so has not only one-sided error, but zero error. Under stronger (but still plausible) assumptions, analogous pseudorandom generators can be made for constant-round public-coin interactive proof systems [AK97, KvM99, BV99]. These generators can be used to derandomize such a proof system by replacing the verifier's messages (which consists of random coin flips) with all possible outputs of the generator. This preserves the prover's complexity and the result is a deterministic proof system (*i.e.,* an $\mathbf{NP}$ proof system), so it certainly has perfect completeness. While these results do not cover all interactive proof systems, they suggest that there may very well be a black-box transformation from interactive proofs to ones with perfect completeness that preserves the prover's complexity. Proposition 12 shows that the existence of such a transformation is closely tied to issues in derandomization; at a minimum it would imply $\mathbf{BPP} = \mathbf{ZPP}$. As we currently only know how to obtain the latter conclusion under intractability assumptions, we would also expect the black-box transformation to rely on such an assumption.

6 Conclusion

This main result of this paper demonstrates that, under standard assumptions, current techniques are insufficient to convert private-coin interactive proofs into public-coin interactive proofs while preserving the complexity of the prover. It would be interesting to give a more absolute separation, showing that it is strictly easier to prove some statements to a private-coin verifier than to a public-coin verifier. That is, construct a problem Π with a private-coin interactive proof (P, V) such that Π has *no* public-coin interactive proof where the prover can be implemented in polynomial time with oracle access to P. Presumably such a result would be under an intractability assumption. Bellare and Goldwasser [BG94] have given results of this nature for a different issue, namely separating the power needed to decide a language from the power needed to prove membership.

While we have considered the problems of converting interactive proofs to ones with public coins or perfect completeness, there are several other general transformations of interactive proofs lacking one or more desirable properties. For example, the transformation from general interactive proofs to zero-knowledge proofs [BGG+88] does not preserve the prover's complexity. Another example is the Collapse Theorem of Babai and Moran [BM88], which does preserve the prover's complexity, but does not preserve any sort of zero knowledge property. Both of these transformations are black-box transformations, and it would be interesting to determine if this makes their deficiencies necessary.

Acknowledgments

This work would not have been possible without the help of Sanjeev Arora, who suggested looking for a result about black-box transformations. I am grateful for his contribution and many illuminating discussions on this topic. I thank Shafi Goldwasser, who sparked my interest in the prover's complexity, and suggested the issues addressed in this paper as research problems. Thanks also to Avi Wigderson for arranging my visit to the Institute for Advanced Study, when much of this work was done, and to Oded Goldreich for all his encouragement.

References

[AK97] V. Arvind and J. Köbler. On resource-bounded measure and pseudorandomness. In *Proceedings of the 17th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. LNCS 1346, Springer-Verlag, 1997.

[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.

[BG94] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, February 1994.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero knowledge. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 494–502, Baltimore, Maryland, 14–16 May 1990.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 1990, 21–25 August 1988.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 113–131, Chicago, Illinois, 2–4 May 1988.

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

[BV99] Peter Bro Miltersen and N.V. Vinodchandran. Derandomizing Arthur–Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science*, New York, NY, 17–19 October 1999. IEEE.

[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.

[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.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 429–442. JAC Press, Inc., 1989.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[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.

[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.

[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.

[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.

[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, August 1999.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, Washington, 15–17 May 1989.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations (extended abstract). In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1988, 16–20 August 1987.

[Kil90] Joe Kilian. Achieving zero-knowledge robustly. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 313–325. Springer-Verlag, 1991, 11–15 August 1990.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 659–667, Atlanta, 1–4 May 1999.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, April 1988.

[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.

[NR97] Moni Naor and Omer Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 189–199, El Paso, Texas, 4–6 May 1997.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[Oka96] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 649–658, Philadelphia, Pennsylvania, 22–24 May 1996. See also preprint of full version, Oct. 1997.

[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.

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