Appears in *Proceedings of the Third Israel Symposium on Theory and Computing Systems*, 1995.

On the Role of Shared Randomness in Two Prover Proof Systems

M. Bellare[*] U. Feige[†] J. Kilian[‡]

January 1995

Abstract

In this paper we consider which aspects of the two prover model are necessary for their striking language recognition and zero-knowledge capabilities.

We approach this question by looking at an alternative, more symmetric model which we call the *double verifier* model. We find that in this model the shared randomness of the verifiers is key to the language recognition power: if the verifiers don't share randomness the power is PSPACE; otherwise it is MIP = NEXPTIME. We find that the shared randomness of the provers is necessary for zero-knowledge: if the provers don't share randomness, statistical zero-knowledge is only possible for languages in $\mathsf{BPP}^{\mathsf{NP}}$; else it is possible for all of NEXPTIME.

These results have immediate implications for the standard two-prover model. We see that correlations between the verifier's queries is crucial for the language recognition power of two prover proofs. In particular, the natural analog of IP = AM does not hold in the two-prover model unless NEXPTIME = PSPACE. Similarly, we see that shared randomness, or correlation of the provers' answers, is necessary for the statistical zero-knowledge of two prover proofs.

[*] Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: mihir@cs.ucsd.edu

[†] Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. e-mail: feige@wisdom.weizmann.ac.il

[‡] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA. e-mail: joe@research.nj.nec.com

1 Introduction

Two prover interactive proofs systems are well known for their language recognition power and potential for zero-knowledge proofs. In this paper we consider which aspects of two prover proof systems are necessary for them to retain these properties. We investigate to what extent two prover proofs may be "simplified" while still retaining the ability to recognize all of NEXPTIME, and moreover to do this in *perfect* zero-knowledge. Our motivation comes from the applications of two prover proofs to approximation theory and to the design of low communication zero-knowledge proofs.

We approach this question by looking at an alternative model, namely the double verifier proof system. This is a clean and "symmetric" version of the two prover model in which the role of various resources can be seen more clearly. We prove a number of results about this model and then translate them back into implications about the standard two prover model.

1.1 The double verifier model

The double verifier (DV) model was suggested by [16]. It "separates" the verifier of the two prover model into a pair of verifiers, one for each prover. Thus the system consists of a pair $V = (V_1, V_2)$ of verifiers and a pair $P = (P_1, P_2)$ of provers. Before the protocol begins the verifiers may share a random string R_V and the provers may share a random string R_P. In addition each party has a source of private coins. (While redundant in the general model, these coins will be important when considering restricted forms of the model). In the interactive phase, V_1 interacts with P_1 to produce transcript t_1; separately, V_2 interacts with P_2 to produce transcript t_2. In these interactions each party computes its messages as a function of the common input x, its shared coins, its private coins, and messages received thus far. For the decision phase, the verifiers reunite and, as a function of x, R_V, t_1, t_2 and their private coins, decide whether to accept or reject.

Note that V_1 and V_2 (resp. P_1 and P_2) cannot communicate during the interactive phase of the protocol. The only link between V_1 and V_2 (resp. P_1 and P_2) is the coins R_V (resp. R_P) they share initially.

Clearly if we allow V_1 and V_2 to communicate during the protocol we recover the two prover model. But, in fact, using the shared coins R_V in lieu of this communication preserves the language recognition power of the model: it follows from [3] that the class DVIP of languages recognized with bounded error by double verifier proof systems equals the class MIP = NEXPTIME (cf. Proposition 2.3). Moreover, it follows from [7] that every language $L \in$ DVIP has a perfect zero-knowledge double verifier proof. Thus, we have found a more "symmetric" version of the two prover model which retains its power.

1.2 Results in the DV model

We ask ourselves a pair of "dual" questions. What happens if we break the symmetry by denying the verifiers the initial shared randomness R_V? Correspondingly, what happens if we deny the provers the initial shared randomness R_P? We find that the shared coins of the verifiers are essential to the language recognition power of the system while the shared coins of the provers are essential to the zero-knowledge.

1.2.1 *The power of independent verifiers*

We say that a pair of verifiers are *independent* if they don't share any initial randomness R_V. That is, their questions and final decision are not functions of R_V. In this case, their ability to verify proofs drops. In fact, having two provers no longer helps, and the power of the system is that of single prover proofs.

Theorem 1.1 L is recognized by a double verifier proof system in which the verifiers are independent if and only if $L \in \mathsf{PSPACE}$.

That is, the class $\mathsf{DVIP}_{\text{iv}}$ of languages recognized (with bounded error) by double verifier proof systems in which the verifiers are independent is just PSPACE.

The fact that $\mathsf{PSPACE} \subseteq \mathsf{DVIP}_{\text{iv}}$ is easy to see (cf. Proposition 2.2). But the other direction, namely $\mathsf{DVIP}_{\text{iv}} \subseteq \mathsf{PSPACE}$, is not obvious, even though it might seem so at first glance. The proof of this theorem is the main technical contribution of this paper and uses the idea of representing a prover strategy by a short "list strategy" (See Section 1.4.3 for a discussion of the related techniques of [18]).

1.2.2 *The power of independent provers*

We say that the provers are *independent* if they don't share any initial randomness R_P. We note that as with other kinds of proof system, the optimal prover strategies in a double verifier proof are deterministic. Thus the shared randomness of the provers is not necessary for language recognition: every $L \in \mathsf{DVIP}$ has a double verifier proof system in which the provers are independent. But independent provers can provide zero-knowledge only for relatively simple languages.

Theorem 1.2 Suppose L has a double verifier, statistical zero-knowledge proof system in which the provers are independent. Then $L \in \mathsf{BPP}^{\mathsf{NP}}$.

Moreover this remains true even if the verifiers are allowed to communicate during the protocol, which will be important in Section 1.3.2.

1.3 Results in the standard model

We now note the implications for the model of two provers communicating with a single verifier [4].

1.3.1 *Two Merlins are no better than one*

In the standard notion of interactive proof systems [13], the verifier's challenges to the prover are based on a set of private coins, and hence may be highly nonrandom and correlated. In Arthur-Merlin games [1, 2] the challenges are independently chosen random strings. But Goldwasser and Sipser [14] showed that the two models have the same language recognition power. A natural question, posed to us by Goldwasser, is whether this equivalence extends to two-prover interactive proof systems. That is, suppose the verifier's questions are purely random. That is, in each round he generates a pair of strings, uniformly and independently distributed both of each other and of all strings in previous rounds. He sends the first string to the first prover and the second string to the second prover. Can such a system recognize $\mathsf{NEXPTIME}$?

A direct corollary of Theorem 1.1 is that these "Two Merlin vs. Arthur" games will recognize no more than the class $\mathsf{IP} = \mathsf{AM} = \mathsf{PSPACE}$ of languages recognized by the usual Arthur-Merlin games. In fact Theorem 1.1 says something stronger, because it only asks that the verifiers be independent: a particular verifier need not send uniformly distributed strings in every round, and need not maintain independence of his queries between rounds.

Viewing Theorem 1.1 in a different perspective, we conclude that correlation between queries to the different provers is an essential ingredient in [3]'s proof that $\mathsf{MIP} = \mathsf{NEXPTIME}$ and cannot be disposed of, unless $\mathsf{PSPACE} = \mathsf{NEXPTIME}$.

We know that two prover proofs can be used to derive intractability of approximation results. Simplifying the model — i.e proving that restricted versions of MIP also equal NEXPTIME — could facilitate reductions to show new combinatorial problems are hard to approximate. In particular, it might have helped if we could assume without loss of generality that the verifier's questions to the two provers are generated independently, and hence are not correlated. Unfortunately, as we saw above, this is impossible unless $\mathsf{PSPACE} = \mathsf{NEXPTIME}$. (This does not mean our results have only negative implications for approximation: see Section 2.3 for more positive results).

1.3.2 *Zero-knowledge needs shared coins*

The standard model for two-prover interactive proof systems allows the provers to agree on a possibly randomized strategy before the protocol begins, or equivalently to share a common random string. Using this shared random string, there exist perfect zero-knowledge proofs for all of NEXPTIME (combine [3] and [4]). In fact, there exist single-round perfect zero-knowledge proofs for all of NEXPTIME with an exponentially small error probability [10]. We note that all of the statistical zero-knowledge interactive proof systems constructed for two-provers use the shared random string in an essential way. It now becomes clear why this is so. We say that the provers in a two prover proof are independent if they don't share any randomness. It then follows from Theorem 1.2 that L has a two prover statistical zero-knowledge proof with independent provers only if $L \in \mathsf{BPP}^{\mathsf{NP}}$. We use here the fact that Theorem 1.2 held also if the verifiers were allowed to communicate during the protocol. Thus shared randomness is necessary for statistical zero-knowledge, at least in the light of our current beliefs on complexity theory.

1.4 Previous and related work

1.4.1 *Simplifications of the two prover model*

Simplifying proof systems, or finding alternative formulations of a proof system which retain properties of interest, is much investigated, both for single and multi prover proofs, perhaps because, using a simpler or alternative form often enables us to obtain results that may otherwise have escaped us. In the multi-prover model we note the following. One cannot eliminate a prover or eliminate the separation of the two provers unless $\mathsf{NEXPTIME} = \mathsf{PSPACE}$. One cannot take away the verifier's random coin tosses, since NP is properly contained in NEXPTIME. There does not exist any nontrivial one-sided proof system in which each prover only sends a bit to the verifier [7], though such proof systems exist if the one-sidedness condition is dropped.

1.4.2 *The role of randomness in ZK proofs*

In the single prover model, it was shown by Goldreich and Oren [12] that removal of the randomness of either party trivializes a zero-knowledge proof system. More specifically, only BPP languages can have single prover (computational) zero-knowledge proofs having either of the following properties: (i) the verifier is deterministic, (ii) the prover is deterministic.

Dwork, Feige, Kilian, Naor and Safra [7] provide some lower bounds on the randomness in two prover, statistical zero-knowledge proofs. Specifically, assuming $\mathsf{BPP} \neq \mathsf{NP}$, they show that in any one-sided statistical zero-knowledge proof system for an NP-complete language either the verifier uses more than $\Omega(\log n)$ random bits or the provers use $\Omega(\log \log n)$ *total* (shared and private) random bits. In contrast, our result shows that *shared* randomness of the provers is necessary for a statistical zero-knowledge proof for any language outside of $\mathsf{BPP}^{\mathsf{NP}}$, regardless of how much private randomness is used by any party. We note that our result also holds for protocols that are only zero-knowledge with respect to an honest verifier; the lower-bound in [7] doesn't hold for this case.

1.4.3 *Techniques*

Our concept of list strategies bares some similarity to a result of Lipton and Young [18] that in two player games, there exist almost optimal mixed strategies which are a combination of a logarithmic number of pure strategies, and hence that there are almost optimal mixed strategy whose representation is logarithmic in the size of the largest mixed strategies. However, in the [18] case, the representation of the near optimal mixed strategy is polynomial in the size of representations of pure strategies, whereas in our case we derive near optimal mixed strategy whose representation is polylogarithmic in the size of pure strategies.

The proof of Theorem 1.2 makes use of the *efficient, almost uniform generation* primitive of [5], and it is this that accounts for the $\mathsf{BPP}^{\mathsf{NP}}$ complexity. The primitive in question is a probabilistic, polynomial time procedure which given an NP oracle can output an element distributed almost uniformly at random in a set. More precisely, given oracles for NP and a set S, and given inputs 1^n and $\delta > 0$, the procedure runs in time $\mathrm{poly}(n, \log \delta^{-1})$, and outputs a random string from a distribution within (statistical) distance δ of the uniform distribution on $S \cap \{0,1\}^n$.[1]

1.4.4 *Related models*

Burmester and Desmedt [6] have also put forth a model of interactive proof systems which features multiple verifiers. In their model, a single prover "broadcasts" to many verifiers. They are primarily concerned with efficiency issues for cryptographic applications.

2 Double Verifier Interactive Proofs

We consider the class of *double verifier proof systems.* Extensions to multiple verifiers are not considered in this paper.

2.1 Definitions

A *verifier* is a polynomial time machine and a *prover* is a computationally unbounded machine. A *double verifier protocol* involves a pair $V = (V_1, V_2)$ of verifiers and a pair $P = (P_1, P_2)$ of provers. Each party has its own, private, coin tosses; the coin tosses of V_i are usually denoted r_i while those of P_i are not made explicit ($i = 1, 2$). Additionally, the verifiers might, before the protocol begins, share a random string R_V and the provers might share a random string R_P. These strings are formally infinite; usually the provers/verifiers look at a prefix of these strings that is of expected size polynomial in the input. The protocol takes place on common input x, whose length will be denoted n. The interaction during the protocol is confined to the following: for each $i = 1, 2$, parties V_i, P_i interact to generate a transcript t_i: this is a random variable over the random choices of R_V, R_P, r_i and the coin tosses of P_i. Then the verifiers evaluate a boolean *decision* predicate on $x, R_V, r_1, t_1, r_2, t_2$ and accept iff the value is 1. We denote by $\mathsf{ACCEPT}_V^P(x)$ the probability that

[1] Note one can attain exponentially small error δ in polynomial time. A primitive for the same task provided in [15] runs in time $\mathrm{poly}(n, \delta^{-1})$, so that one can only attain polynomially small error in polynomial time: this won't suffice for our application.

the verifiers accept in this interaction; the probability is over R_V, r_1, r_2 and the coins (shared and private) of the provers. We let $\mathsf{ACCEPT}_V(x)$ be the maximum of $\mathsf{ACCEPT}_V^P(x)$ over all pairs of provers P.

Observe that in double verifier interactive protocols, V_1 does not communicate with V_2, and P_1 does not communicate with P_2. Of course V_i does not communicate with P_j for $i \neq j$. Also, r_1 and r_2 are independently distributed. Had we allowed communication between the verifiers we would recover the two prover model. Had we additionally allowed communication between the provers we would recover the single prover model.

We say the verifiers are *independent* if their messages (each to its respective prover) are not functions of R_V. More precisely, we require that they run in polynomial time without having R_V as an input; we don't know of any example where this technical distinction is important. We say that the provers are *independent* if their strategies are not functions of R_P.

Definition 2.1 Let $0 < \epsilon_\ell(\cdot) \leq \epsilon_h(\cdot) \leq 1$. Let $V = (V_1, V_2)$ be a pair of verifiers and $P = (P_1, P_2)$ a pair of provers. We say that P, V is a double verifier interactive proof system with error $(\epsilon_h, \epsilon_\ell)$ for language L if for every x:

(1) Completeness: $x \in L$ implies $\mathsf{ACCEPT}_V^P(x) \geq \epsilon_h(|x|)$, and

(2) Soundness: $x \notin L$ implies $\mathsf{ACCEPT}_V(x) \leq \epsilon_\ell(|x|)$.

When the error has the form $(1-\epsilon, \epsilon)$ for some ϵ then we abuse terminology and say that the proof system has error ϵ. L has a double verifier proof if it has a double verifier proof with error $\epsilon \leq 1/3$. The error can be reduced by standard means.

DVIP denotes the class of all languages possessing double verifier proofs. $\mathsf{DVIP}_{\mathrm{iv}}$ denotes the class of all languages possessing double verifier proofs in which the verifiers are independent.

A double verifier proof system is said to have unbounded error if it has error $(\epsilon_h, \epsilon_\ell)$ with $\epsilon_\ell(n) < \epsilon_h(n)$ for all n. $\mathsf{DVIP}^{\mathrm{u}}$ denotes the class of all languages possessing double verifier proofs with unbounded error and $\mathsf{DVIP}_{\mathrm{iv}}^{\mathrm{u}}$ denotes the class of all languages possessing double verifier proofs in which the verifiers are independent and the error is unbounded.

2.2 Some elementary properties of the model

It is known that for both single and two prover interactive proofs the class of recognized languages remains the same whether one asks for bounded or unbounded error. Below, we see that the same is true of double verifier proofs. But, interestingly, we will see that the type of error makes a difference when one makes the verifiers independent.

We begin by observing that if $L \in \mathsf{PSPACE}$ then L has a double verifier interactive proof in which the verifiers are independent.

Proposition 2.2 $\mathsf{PSPACE} \subseteq \mathsf{DVIP}_{\mathrm{iv}}$.

Proof: The standard model of one prover interactive proofs [13, 2] is a special case of double verifier interactive proofs. In this special case, the predicate $\mathsf{accept}(x, R_V, r_1, t_1, r_2, t_2)$ ignores the entries R_V, r_2 and t_2. The result now follows from [21, 19]. ▌

When the verifiers make use of their shared randomness they can recognize all of $\mathsf{NEXPTIME}$. Moreover, this is true for both the case of bounded error and the case of unbounded error.

Proposition 2.3 $\mathsf{DVIP} = \mathsf{DVIP}^{\mathrm{u}} = \mathsf{NEXPTIME}$.

Proof: Since $\mathsf{DVIP} \subseteq \mathsf{DVIP}^{\mathrm{u}}$ it suffices to show that $\mathsf{DVIP}^{\mathrm{u}} \subseteq \mathsf{NEXPTIME}$ and $\mathsf{NEXPTIME} \subseteq \mathsf{DVIP}$. For the first, one can use the argument that $\mathsf{MIP} \subseteq \mathsf{NEXPTIME}$, noting that it does indeed apply even when the error is unbounded. On the other hand, the protocol of [3] for NEXPTIME does not require the queries to one prover to depend on the answers of the other prover, and hence can be executed in the double verifier model. Details are omitted. ▮

Hence the double verifier model is as powerful as general multiprover models as long as we allow the verifiers to share coins. (It is instructive to think of how the verifiers run the protocol of [3] in the above and note that they *do* use their shared coins).

When only a single round of interaction is allowed, it is easy to see that the standard two prover model and the double verifier model are the same. Thus results in the former translate to the latter. In particular, we have the following.

Proposition 2.4 Every $L \in \mathsf{NEXPTIME}$ has a bounded error, perfect zero-knowledge, double verifier proof system with one round and exponentially small error.

Proof: Follows from [10] which in turn built on [17, 11, 7]. Details are omitted. ▮

When we disallow the shared coins, the language recognition power is unchanged for the unbounded error case:

Proposition 2.5 $\mathsf{DVIP}^{\mathrm{u}}_{\mathrm{iv}} = \mathsf{NEXPTIME}$.

Proof: The verifiers follow a [3] type proof system for NEXPTIME as indicated above, with V_i using r_i in the role of the shared coins. When they get together, they reject if $r_1 \neq r_2$. Else they accept iff the original protocol indicated that they should accept. Thus if $x \in L$ then $\mathsf{ACCEPT}_V(x) = 2^{-l}$ where $l = |r_1| = |r_2|$, and if $x \notin L$ then $\mathsf{ACCEPT}_V(x) \leq \epsilon \cdot 2^{-l}$ where ϵ is the error probability of the original system. ▮

In contrast, Theorem 1.1 says that the language recognition power for the bounded error case drops to PSPACE.

2.3 Implications for approximation

Results on the language recognition power of interactive proof systems can be turned around and presented as hardness results on the complexity of approximating the maximum attainable success probability of the provers. The motivation is that thereafter, reductions can be used in order to show hardness of approximation results for other problems, following the paradigm of [9]. In this context, parameters of the proof system are usually scaled down, so as to obtain NP-hardness results rather than the less interesting NEXPTIME-hardness results. In this context an implication of the difference between bounded and unbounded error for double *independent* verifiers is given by the following proposition. It says that the accepting probability of independent verifiers can be efficiently approximated to within a small additive error, but not within even large multiplicative factors.

Proposition 2.6

(1) Let $\gamma(n) \geq 1/\mathrm{poly}(n)$ and let $V = (V_1, V_2)$ be independent. Then there is a function A, computable in polynomial space, such that $|A(x) - \mathsf{ACCEPT}_V(x)| \leq \gamma(|x|)$ for all x.

(2) For any $\gamma(n) \leq 2^{\mathrm{poly}(n)}$ there is an independent $V = (V_1, V_2)$ such that approximating $\mathsf{ACCEPT}_V(\cdot)$ within a multiplicative factor of $\gamma(|x|)$ is NEXPTIME-hard.

Proof: The first item is a consequence of the proof, given in Section 3, that $\mathsf{DVIP}_{\text{iv}} \subseteq \mathsf{PSPACE}$. For the second item, use the proof of Proposition 2.5 and notice that the initial ϵ can be made exponentially small. We omit the details. ▮

The context of approximation sheds more light on the effect of allowing or not allowing the verifiers to share a common random string. When the verifiers do share a random string the acceptance probability exhibits a gap which, in the terminology of [20], is at location 1. Removing the shared random string moves the gap location to something very close to 0. But in both cases the ratio between the boundaries of the gap is exponential, so that approximating the acceptance probability within a multiplicative, exponential factor is NEXPTIME-hard.

Item (2) of Proposition 2.6 might be useful for proving non-approximability results, but, given that its proof follows in a simple way from related results in the standard two prover model, we don't a priori expect there to be a substantial gain over using the standard two prover model directly in this regard.

3 Independent verifiers recognize exactly PSPACE

In this section we prove Theorem 1.1. We already saw that $\mathsf{PSPACE} \subseteq \mathsf{DVIP}_{\text{iv}}$ (Proposition 2.2). Now suppose $L \in \mathsf{DVIP}_{\text{iv}}$. We want to show that $L \in \mathsf{PSPACE}$.

3.1 Overview of the proof

Let (P_1, P_2, V_1, V_2) be a proof system for L, where P_1 and P_2 are deterministic optimal provers for $V = (V_1, V_2)$. If the optimal strategies for P_1 and P_2 could be computed in PSPACE, then membership in L could be computed in PSPACE. While these optimal strategies may be NEXP-hard, we can transform (P_1, P_2, V_1, V_2) into a proof system, (P_1', P_2', V_1, V_2) with the following properties,

(1) The strategies for P_1' and P_2' are deterministically computable in PSPACE.

(2) If $x \in L$, then $\mathsf{ACCEPT}(P_1, P_2, V_1, V_2, x) - \mathsf{ACCEPT}(P_1', P_2', V_1, V_2, x) \leq \epsilon(x)$, where $\epsilon(x) > 1/|x|^{O(1)}$.

Thus, the transformed provers are PSPACE-bounded, but almost as effective as the original provers. In particular, there is an easily computable threshold λ such that $\text{accept}(P_1', P_2', V_1, V_2, x) \geq \lambda$ whenever $x \in L$, and $\text{accept}(P_1', P_2', V_1, V_2, x) < \lambda$ whenever $x \notin L$. These two cases may be distinguished in PSPACE by running (P_1', P_2', V_1, V_2) for every possible value of V_1 and V_2's coin tosses, yielding a PSPACE decision procedure for L. It remains to show how to construct near-optimal provers that run in PSPACE.

We assume without loss of generality that V enforces some limit $\ell(x)$ on the total size of (r_1, t_1, r_2, t_2). In our construction of efficient near-optimal provers, we first show how to approximate one of the original prover-verifier pairs by a much simpler object: a family of transcript-lists. Notationally, we replace (P_1, P_2, V_1, V_2) by $(P_1, V_1, \{T_x^1\})$. For each x, T_x^1 contains a short list of transcripts. Instead of combining its results with the interaction between V_2 and P_2, V_1 simply chooses a transcript at random from T_x^1, and acts as if it was the (P_2, V_2) transcript. Here our notion of transcripts includes V_2's private random string as well as V_2's conversation with P_2. In general, this transformation does not yield any meaningful result for any short set of transcripts. However, when the verifiers are independent, we show that there exists $\{T_x^1\}$ such that for all P_1^* and x, (P_1^*, P_2, V_1, V_2) and $(P_1^*, V_1, \{T_x^1\})$ accept x with nearly the same probability.

We can view $(P_1^*, V_1, \{T_x^1\})$ as a single prover proof system that takes as input (x, T_x^1), and therefore there is an optimal prover $P_1^*[T_x^1]$ that runs in PSPACE, but must take T_x^1 as input as well as x. By the approximation result, $(P_1^*[T_x^1], P_2)$ will perform nearly as well as (P_1, P_2). We then approximate $(P_1^*[T_x^1], P_2, V_1, V_2)$ by $(\{T_x^2\}, P_2, V_2)$ and construct a PSPACE-bounded optimal prover $P_2^*[T_x^2]$. $(P_1^*[T_x^1], P_2^*[T_x^2])$ will perform nearly as well as (P_1, P_2), and hence if (P_1, P_2) is optimal, then $(P_1^*[T_x^1], P_2^*[T_x^2])$ will be near-optimal. Finally, we show that a suitable T_x^1 and T_x^2 may be computed in PSPACE, yielding truly PSPACE-bounded provers P_1' and P_2'.

3.2 Converting (P_1, P_2, V_1, V_2) to $(P_1, V_1, \{T_x^1\})$

Consider the probabilistic function $\mathsf{accept}'(x, r_1, t_1)$ that generates r_2 and t_2 according to V_2 and P_2 and then outputs the value of $\mathsf{accept}(r_1, t_1, r_2, t_2)$. One can consider the protocol to be executed as follows:

(1) V_1 generates a random string r_1. Using r_1 and x, V_1 interacts with P_1, generating a transcript t_1.

(2) V_1 samples $\mathsf{accept}'(x, r_1, t_1)$, and accepts or rejects accordingly.

If this procedure accepts x with probability p then (P_1, P_2, V_1, V_2) also accepts x with probability p. Thus, we have reduced the system to one with a single prover, but computing accept' may be as hard as computing P_2's strategy, so the verifier may no longer run in polynomial time.

We approximate P_2 by a family of lists $\{T_x^1\}$. To show that the approximation makes sense, we exhibit a probabilistic function accept^* that approximates accept', but is easily computable given T_x^1.

Let $T = \{(r_{2,1}, t_{2,1}), \ldots, (r_{2,m}, t_{2,m})\}$. We define $\mathsf{accept}^*(x, r_1, t_1, T)$ as the probabilistic function computed by uniformly choosing $1 \leq i \leq m$ and returning the value of $\mathsf{accept}(x, r_1, t_1, r_{2,i}, t_{2,i})$.

One can think of T as a list of "canned" transcripts. Choosing a transcript from T at random approximates choosing a transcript from the interaction between V_2 and P_2. Lemma 3.1 shows that there exists a polynomial-sized T_x^1 such that $\mathsf{accept}^*(x, r_1, t_1, T_x^1)$ is a uniformly good approximation for $\mathsf{accept}'(x, r_1, t_1)$.

Lemma 3.1 Let $\epsilon(x) > 1/|x|^{O(1)}$ and let $\ell(x)$ denote the maximum possible size of (r_1, t_1, r_2, t_2). Then for all x there exists T_x^1 such that

- $|T_x^1| < (\ell/\epsilon(x))^c$, for some global constant c, and
- For all (r_1, t_1) it is the case that

$$\begin{aligned}|\Pr[\mathsf{accept}^*(x, r_1, t_1, T_x^1) = 1] \\ -\Pr[\mathsf{accept}'(x, r_1, t_1) = 1]| < \epsilon(x).\end{aligned}$$

Remark: The lemma holds for all $\epsilon(x) > 0$, but we make the added restriction to emphasize the point that $|T_x^1|$ can be made polynomial in $|x|$.

Proof of Lemma 3.1: We show that a sufficiently large randomly chosen T_x^1 will suffice with high probability. Fix r_1 and t_1, and let p denote the probability that $\mathsf{accept}'(x, r_1, t_1) = 1$. Let T_x^1 be generated as follows. For m sufficiently large and for $1 \leq i \leq m$, choose $r_{2,i}$ uniformly and independently, and let $t_{2,i}$ by the result of running V_2 and P_2 on x, using $r_{2,i}$ as V_2's random input. (Note since P_2 is deterministic and fixed, $t_{2,i}$ is determined by $r_{2,i}$). Let $X_i^{r_1} = \mathsf{accept}(x, r_1, t_1, r_{2,i}, t_{2,i})$. Note $X_1^{r_1,t_1}, \ldots, X_m^{r_1,t_1}$ are independent random variables, each with expectation p. Let $X^{r_1,t_1} = X_1^{r_1,t_1} + \ldots + X_m^{r_1,t_1}$. We note that

$$\Pr[\,\mathsf{accept}^*(x, r_1, t_1, T_x) = 1\,] \;=\; \tfrac{1}{m} X^{r_1,t_1}\ .$$

Using standard Chernoff bounds,

$$\Pr\left[\,\left|\tfrac{X^{r_1,t_1}}{m} - p\right| > \epsilon\,\right] \;\leq\; e^{-\epsilon^2 m/2}\ ,$$

and for large enough m this can be made strictly less than $2^{-\ell(x)}$. So

$$\Pr\left[\,\exists r_1, t_1\ :\ \left|\tfrac{X^{r_1,t_1}}{m} - p\right| > \epsilon\,\right] \;<\; 1\ .$$

Hence, some T_x^1 meeting the conditions of the lemma must exist. ▮
Consider the proof system $(P_1, V_1, \{T_x^1\})$ that works as follows. On input x, V_1 interacts with P_1 as before, but instead of having V_2 and P_2 interact, (r_2, t_2) is generated by choosing at random from T_x^1. V_1 then accepts or rejects as before. We define $\mathsf{ACCEPT}(P_1, V_1, \{T_x^1\}, x)$ to be the probability that $(P_1, V_1, \{T_x^1\})$ accepts on input x. Lemma 3.2 states that if $\{T_x^1\}$ is chosen according to Lemma 3.1, then for every P^* it is the case that $(P^*, V_1, \{T_x^1\})$ is a good approximation to (P_1, P_2, V_1, V_2).

Lemma 3.2 Let $\epsilon(x)$ and T_x^1 be as in Lemma 3.1. Then for all P^* it is the case that $|\mathsf{ACCEPT}(P^*, P_2, V_1, V_2, x) - \mathsf{ACCEPT}(P^*, V_1, \{T_x^1\}, x)| < \epsilon(x)$.

Proof: Let $\rho(r_1, t_1)$ be the probability that P^* and V_1 generate (r_1, t_1). By a straightforward probability argument, we have

$$\begin{aligned} &\mathsf{ACCEPT}(P^*, P_2, V_1, V_2, x) \\ &\quad = \ \textstyle\sum_{(r_1,t_1)} \rho(r_1, t_1) \mathsf{Pr}[\mathsf{accept}'(x, r_1, t_1) = 1] \end{aligned}$$

and

$$\begin{aligned} &\mathsf{ACCEPT}(P^*, \{T_x^1\}, V_1, x) \\ &\quad = \ \textstyle\sum_{(r_1,t_1)} \rho(r_1, t_1) \mathsf{Pr}[\mathsf{accept}^*(x, r_1, t_1, T_x^1) = 1]. \end{aligned}$$

Thus, we can bound $|\mathsf{ACCEPT}(P^*, P_2, V_1, V_2, x) - \mathsf{ACCEPT}(P^*, \{T_x^1\}, V_1, x)|$ from above by

$$\textstyle\sum_{(r_1,t_1)} \rho(r_1, t_1) \cdot \delta(x, r_1, t_1, T_x^1)$$

where we have set $\delta(x, r_1, t_1, T_x^1) = |\mathsf{Pr}[\mathsf{accept}'(x, r_1, t_1) = 1] - \mathsf{Pr}[\mathsf{accept}^*(x, r_1, t_1, T_x^1) = 1]|$. But this quantity is at most $\epsilon(x)$. So the bound is $\epsilon(x)$ as desired. ▮

3.3 Constructing near-optimal provers using advice

Lemma 3.3 states that an optimal prover for convincing V_1 in conjunction with $\{T_x^1\}$ is a near-optimal prover for convincing (V_1, V_2) in conjunction with P_2.

Lemma 3.3 Let P_2 and V_2 be fixed, and let $\epsilon(x)$ and T_x^1 be as in Lemma 3.1. Let $P_1^*[T_x^1]$ be an optimal prover for convincing V_1 to accept x in conjunction with $\{T_x^1\}$. Then for all P_1 it is the case that $a - b < 2\epsilon(x)$ where

$$\begin{aligned} a &= \mathsf{ACCEPT}(P_1, P_2, V_1, V_2, x) \\ b &= \mathsf{ACCEPT}(P_1^*[T_x^1], P_2, V_1, V_2, x)\ . \end{aligned}$$

Proof: We let

$$\begin{aligned} c &= \mathsf{ACCEPT}(P_1, V_1, \{T_x^1\}, x) \\ d &= \mathsf{ACCEPT}(P_1^*[T_x^1], V_1, \{T_x^1\}, x)\ . \end{aligned}$$

By Lemma 3.2, we have

$$\begin{aligned} a-c &< \epsilon(x) \\ d-b &< \epsilon(x)\ . \end{aligned}$$

But since $P_1^*[T_x^1]$ is optimal, $d \geq c$. Combining these inequalities gives the bound stated in the Lemma. ▮

Using the above lemmas, we show the existence of a pair of provers which are near-optimal and can be computed in PSPACE given access to an advice string for each input.

Lemma 3.4 Let (P_1, P_2, V_1, V_2) be an independent-query two-prover proof system, and let $\epsilon(x) > 1/|x|^{O(1)}$. There exist provers P_1^* and P_2^* and families of transcript lists $\{T_x^1\}$ and $\{T_x^2\}$ such that
(1) P_i^* runs in polynomial space on input (x, T_x^i), and
(2) $a - f < \epsilon(x)$ where

$$\begin{aligned} a &= \mathsf{ACCEPT}(P_1, P_2, V_1, V_2, x) \\ f &= \mathsf{ACCEPT}(P_1^*[T_x^1], P_2^*[T_x^2], V_1, V_2, x)\ . \end{aligned}$$

Proof: By Lemmas 3.1–3.3, we can construct a prover $P_1^*[T_x^1]$ such that $a - b < \epsilon(x)/2$, where

$$b = \mathsf{ACCEPT}(P_1^*[T_x^1], P_2, V_1, V_2, x)\ .$$

Performing the same construction on $(P_1^*[T_x^1], P_2, V)$, this time with the second prover, we obtain a prover $P_2^*[T_x^2]$ such that $b - f < \epsilon(x)/2$. Combining these inequalities gives $a - b < \epsilon(x)$. Now each P_i^* is a single prover trying to convince a polynomial time verifier in conjunction with T_x^i. But this is equivalent to a single-prover proof system with input (x, T_x^i). Hence, by the result of [8] there exists an optimal $P_i^*[T_x^i]$ that is computable in PSPACE on input (x, T_x^i). ▮

3.4 Eliminating the advice

To complete the proof of Theorem 1.1, we show that useful (T_x^1, T_x^2) can be generated in PSPACE from input x. First, we note that Feldman's proof that PSPACE-bounded optimal provers exist is constructive. That is, given V, the input (x, T_x^1, T_x^2) and the transcript of the conversation so far, there exists a PSPACE-bounded procedure that will produce $P_i^*[T_x^i]$'s next response. Thus, given the (T_x^1, T_x^2) we can compute the probability that the constructed $(P_1^*[T_x^1], P_2^*[T_x^2])$ will cause V to accept in PSPACE. Hence, by exhaustive search, the modified provers P_1' and P_2' can compute the optimal (T_x^1, T_x^2) in PSPACE and then simulate $P_1^*[T_x^1]$ and $P_2^*[T_x^2]$. The theorem follows.

4 Shared random strings and statistical zero-knowledge

Here is a sketch of the proof of Theorem 1.2. We in fact prove this result for the somewhat stronger case of a standard verifier instead of a double verifier. Thus we consider a standard two prover (and single verifier) proof system but with the provers independent. Suppose L is accepted by such a system (P_1, P_2, V). As before, we construct a more efficient pair of provers (P_1', P_2') such that (P_1', P_2', V) is also a proof system. In this case, P_i''s strategy can be computed in probabilistic polynomial time given access to an oracle for NP (technically, an oracle for any NP-complete language). To decide L in $\mathsf{BPP}^{\mathsf{NP}}$ we can then run the interactive proof (P_1', P_2', V), and accept or reject according to the decision of the verifier.

To construct efficient provers (P_1', P_2'), the first observation we make is that each prover's response is dependent on his conversation thus far with the verifier, and is independent of the messages sent by the other prover. Note that this is not true with an arbitrary two-prover proof system, because here the provers may share coins.

Given a distribution D, let $\omega \leftarrow D$ denote the operation of sampling an element from D. Let $\mathcal{P}$ be a predicate such that $\mathcal{P}(\omega)$ holds with nonzero probability when $\omega \leftarrow D$. We denote by $\omega \overset{\mathcal{P}}{\leftarrow} D$ the operation of repeatedly taking samples $\omega \leftarrow D$ until $\mathcal{P}(\omega)$, and then returning ω. Finally, we denote by $(P_1, P_2, V)(x)$ the distribution on the conversations generated by (P_1, P_2, V) on input x.

We use the efficient, almost uniform generation primitive of [5]. Suppose that D is samplable in polynomial time, and $\mathcal{P}$ is computable in polynomial time. Then if one is given an oracle for NP, the primitive in question allows $\omega \overset{\mathcal{P}}{\leftarrow} D$ to be approximately sampled in probabilistic polynomial time. To be more precise, the primitive will generate a string distributed at random in a distribution which has statistical distance δ from the desired one, while running in time $\mathrm{poly}(\log \delta^{-1})$. This means δ can be made exponentially small with polynomial overhead. Hence, for this sketch we will treat it as a perfect sampling procedure.

We can view a prover as a probabilistic function that takes a partial transcript representing its conversation thus far with the verifier, and returns a response. We define P_1^* as follows (P_2^* is defined analogously). Given the partial transcript t of P_1^*'s interaction with V, let $\mathcal{P}_h(\omega)$ denote the predicate that holds iff ω agrees with t. P_1^* samples $\omega \overset{\mathcal{P}_h}{\leftarrow} (P_1, P_2, V)(x)$ and gives as his response the response given by ω. The key observation is that P_1^* has the same functional behavior as P_1. This follows from the definition of $(P_1, P_2, V)(x)$ and the independence property of the provers.

We now show how to implement P_1^* in polynomial time with an NP oracle. First, suppose that the simulator S for (P_1, P_2, V) works in polynomial time (in general, S may work in expected polynomial time). By the definition of zero-knowledge, sampling $\omega \leftarrow (P_1, P_2, V)(x)$ is equivalent to sampling $\omega \leftarrow S(x)$ whenever $x \in L$, and hence sampling $\omega \overset{\mathcal{P}_h}{\leftarrow} (P_1, P_2, V)(x)$ is equivalent to sampling $\omega \overset{\mathcal{P}_h}{\leftarrow} S(x)$. Thus, using the result of [5], P_1^* can be implemented in the desired time-bound.

When $S(x)$ is only guaranteed to run in expected polynomial time, the proof is greatly complicated. We first consider the approximate simulator $S_q(x)$, where q is a polynomial. $S_q(x)$ runs S, but aborts if S attempts to run for longer than $q(|x|)$ steps. We can use the above procedure to efficiently (given an NPoracle) implement approximate P_1^* and P_2^* based on S_q, and call the resulting provers P_1' and P_2'. Note that unlike P_1^* and P_2^*, P_1' and P_2' will no longer implement the same strategies as P_1 and P_2. However, we inductively show that when one runs (P_1', P_2', V) on input x then most of the time the simulated strategy will be reasonably close to the original strategy. The distribution of complete transcripts (the complete conversation along with V's coin-tosses) generated by (P_1', P_2', V) on input x will approximate that generated by (P_1, P_2, V) in the following sense: For all but $|x|^{-c}$ of the probability mass of the complete transcripts, the probability assigned by (P_1', P_2', V) to each complete transcript is to within a $(1 \pm |x|^{-c})$ multiplicative factor of the probability assigned it by (P_1, P_2, V). Here, c can be made as large as desired, provided that q is set to be a sufficiently large polynomial. This implies that if (P_1, P_2, V) accepts x with probability p, then (P_1', P_2', V) accept x with probability at least $(1 - |x|^{-c})(p - |x|^{-c})$. Thus, (P_1', P_2') serve as efficient near-optimal provers, which implies the theorem.

Acknowledgments

Work done while the first author was at the IBM T.J. Watson Research Center, New York.

References

[1] L. Babai. Trading group theory for randomness. *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing,* ACM, 1985.

[2] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *J. Computer and System Sciences* **36**, 254–276, 1988.

[3] L. Babai, L. Fortnow and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[4] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

[5] M. Bellare and E. Petrank. Making Zero-Knowledge Provers Efficient. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[6] M. Burmester and Y. Desmedt. Broadcast interactive proofs. *Advances in Cryptology – Eurocrypt 91 Proceedings,* Lecture Notes in Computer Science Vol. 547, Springer-Verlag 1991.

[7] C. Dwork, U. Feige, J. Kilian, M. Naor and M. Safra. Low communication 2-prover zero-knowledge proofs for NP. *Advances in Cryptology – Crypto 92 Proceedings,* Lecture Notes in Computer Science Vol. 740, Springer-Verlag, E. Brickell, ed., 1993.

[8] P. Feldman. The optimal prover lives in PSPACE. Manuscript.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science*, IEEE, 1991.

[10] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[11] U. Feige, L. Lovasz. Two-prover one-round proof systems, their power and their problems. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[12] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology* Vol. 7, No. 1, 1994.

[13] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *SIAM J. Comput.* **18** (1), 186-208, February 1989.

[14] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research (ed. S. Micali)* Vol. 18, 1989.

[15] R. Impagliazzo, L. Levin and M. Luby. Pseudo-Random Generation from One-Way Functions. *Proceedings of the Twenty First Annual Symposium on the Theory of Computing,* ACM, 1989.

[16] J. Kilian. Strong separation models of multi-prover interactive proofs. *Analog Science Fiction*, to appear.

[17] D. Lapidot and A. Shamir. Fully Parallelized Multi-prover protocols for NEXP-time. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[18] R. Lipton and N. Young. Simple Strategies for Large Zero-Sum Games with Applications to Complexity Theory. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[19] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[20] E. Petrank. The hardness of approximation: Gap location. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993.

[21] A. Shamir. IP=PSPACE. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.