Low communication 2-prover zero-knowledge proofs for NP

(Preliminary Version)

Cynthia Dwork[1] Uri Feige[2] Joe Kilian[3] Moni Naor[1] Muli Safra[1]

[1] IBM Research Division, Almaden Research Center
[2] IBM Research Division, T. J. Watson Research Center
[3] NEC Research

Abstract. We exhibit a two-prover perfect zero-knowledge proof system for 3-SAT. In this protocol, the verifier asks a single message to each prover, whose size grows logarithmically in the size of the 3-SAT formula. Each prover's answer consists of only a constant number of bits. The verifier will always accept correct proofs. Given an unsatisfiable formula S the verifier will reject with probability at least $\Omega((|S|-\textbf{max-sat}(S))/|S|$, where $\textbf{max-sat}(S)$ denotes the maximum number of clauses of S that may be simultaneously satisfied, and $|S|$ denotes the total number of clauses of S. Using a recent result by Arora et al [2], we can construct for any language in NP a protocol with the property that any non-member of the language be rejected with constant probability.

1 Introduction

In a multiple-prover interactive proof system, several provers, $P_1, P_2, \ldots$ try to convince a verifier V that a common input x belongs to a language L. The verification proceeds in rounds; in each round, the verifier sends to each prover a private message (query) and receives an answer. Each prover sees only the queries addressed to it, and cannot communicate with the other provers (at least until the end of the round). When the protocol ends, the verifier decides, based on the input string and the messages received, whether or not to accept.

Multi-prover proof systems were introduced by Ben-Or, Goldwasser, Kilian and Wigderson [7] in order to obtain zero knowledge proofs without relying on complexity assumptions such as the existence of one-way functions. In this paper we show another advantage of multi-prover proof systems by exhibiting a low communication two-prover perfect zero-knowledge proof system for 3-SAT (and thus for every language in NP). In contrast, no such low communication zero knowledge protocol is possible in a *single* prover proof system, unless $NP \subset BPP$.

Kilian [16] has provided additional motivation for striving for low communication in the two prover setting: he suggests enforcing the separation of the two provers by keeping them (say the two provers are implemented on a smart

E.F. Brickell (Ed.): Advances in Cryptology - CRYPTO '92, LNCS 740, pp. 215-227, 1993.

card) at some distance from each other. If the distance is long enough and the communication complexity is low, then the two provers do not have enough time to communicate during the execution of the protocol.

In the protocol we present, the verifier sends to each of the two provers a query whose length is logarithmic in the length of the input string, and receives back answers whose length is constant. If the input string is not in the language, then the verifier detects cheating with some fixed probability $\alpha > 0$. The protocol is perfect zero-knowledge, i.e. there is a polynomial time machine, called the simulator, that produces for every possible (possibly cheating) verifier the same distribution of conversations as the verifier would have had with two "real" provers.

To reduce the probability of error to 2^{-k} (rather than $1 - \alpha$), the protocol can be executed $O(k)$ times sequentially. Lapidot and Shamir [18] have provided an elegant zero-knowledge two prover protocol which is parallelizable, i.e. running copies of it in parallel decreases the probability of error exponentially in the number of copies. However, it is not known whether this is true for general protocols. Feige and Lovasz [13] (continuing [19]) have provided a method that can be applied to any protocol in order to obtain a parallelizable protocol, however the method does not preserve zero-knowledge. Finding such a method that preserves zero-knowledge is an open question.

In our protocol the two provers share a common random string of only logarithmic length. Thus, even if we consider the shared random string to be part of the communication complexity of the protocol, then it is still logarithmic. The existence of a shared random string is necessary, since we show that for low communication zero-knowledge protocols, the only languages that do not require the two provers to share a common random string are exactly those in BPP.

Our protocol is constructive in the sense that once two provers know a satisfying assignment to the formula, all they are required to do is some polynomial time computation.

1.1 Definitions

Definition 1 We say that a language L has a *two prover interactive proof system* if there exists an interactive probabilistic polynomial time machine (called the verifier) V and two interactive machines P_1, P_2 called Prover 1 and Prover 2 respectively, satisfying the following conditions. All three machines have a common input x which may or may not be in L. The two provers once and for all agree on a common strategy. Moreover, prior to each execution of the protocol, they may interact in order to share random bits. Once the protocol begins, they are assumed to be isolated from each other. The three machines follow a prescribed protocol consisting of several rounds; in each round, the verifier sends to each prover in private a message (query) and receives an answer. When the protocol ends, the verifier decides whether or not to accept, based on the input string and the messages received. The protocol must satisfy

- $\forall x \in L$ there exist machines P_1 and P_2 such that V accepts with probability 1 (completeness);
- there is a fixed constant $\alpha > 0$ such that $\forall x \notin L$ and $\forall P_1, P_2$ the probability that V accepts on input x is at most $1 - \alpha$.

Note that this definition is not standard in that α is not required to be say 2/3. However, by running the protocol *sequentially* several times (as a function of α) one can get arbitrary small probability of accepting erroneously. Showing that the probability goes down when the protocols are run in parallel is a major open problem in this area.

Part of the strategy that the two provers agree on may simply be a common random string. This is used to obtain the zero-knowledge property defined below.

Definition 2 For a given verifier V, provers P_1 and P_2, and input x, we define $View_{V,P_1,P_2}(x)$ be the distribution over the interaction between verifier V and provers P_1 and P_2. This distribution is over V's coin tosses and the random choices made by P_1 and P_2

Definition 3 A two prover interactive protocol V, P_1, P_2 is *perfect zero knowledge for* V if there exists a probabilistic polynomial time machine S that on input x outputs a string whose distribution is $View_{V,P_1,P_2}(x)$. A language L is said to have a perfect zero-knowledge protocol if it has a two-prover interactive proof system V, P_1, P_2 such that for every V' the protocol V', P_1, P_2 is perfect zero-knowledge for V'.

The communication complexity of a protocol is composed of three parts:

1. the total length of the queries sent by the verifiers;
2. the total length of the answers given by the provers;
3. the length of the random string shared by the two provers.

The term *low communication* will mean that the sum of these three components is logarithmic in the length of the input string.

1.2 Background

Multi-prover proof systems have inspired much research in Complexity Theory [5, 8, 9, 10, 11, 13, 14, 19]. In particular, Babai, Fortnow and Lund have shown that the class of languages that are recognized by multiple prover proof system where the verifier is a polynomial time machine and the communication is restricted to be of polynomial length is exactly NEXP-Time. This was scaled down to the NP setting [3, 4, 12], culminating in the result of Arora, Lund, Motwani, Sudan and Szegedy [2] showing a two prover proof system for NP in which the length of the queries that the verifier sends to the provers is logarithmic in the length of the input string, and the answers are of constant length. From this they derive:

Theorem 1 *[2] There is a $\beta > 0$ such that for any language $L \in NP$ there is a polynomial time reduction R from L to 3-CNF formulas such that for $x \in L$ $R(x)$ is a satisfiable 3-CNF and for all $x \notin L$, a fraction of at most $1 - \beta$ of the clauses of $R(x)$ can be satisfied simultaneously. The proof is constructive in the sense that given a witness for x's membership in L, there is a polynomial time procedure that yields a satisfying assignment to $R(x)$.*

We will apply this theorem to get our protocols. This theorem (or actually its precursor [4]) was already used by Kilian [17] to lower the communication complexity of single prover zero knowledge arguments and proof systems. However, by a simple observation, the only languages that have a single prover proof system with logarithmic communication are those in BPP. Thus, if we are aiming at logarithmic communication we must have two provers.

We further observe that the two provers must share a random string in order for a low-communication protocol to be zero-knowledge; for if not, by running the simulator enough times we can get the response on any query to each prover, and thus can simulate each prover *on-line*. If the two provers do not share a random string, then their responses are independent polynomial time samplable distributions and thus there is a probabilistic polynomial time machine that can compute the probability that the verifier accepts, whence L is in BPP. We do not know whether it is possible for the two provers to share fewer than the logarithmically many random bits required by our protocol. However, in Section 3 we show that $\Omega(\log \log n)$ random bits are essential.

2 The Interactive Proof System

We construct the interactive proof system in two steps. In the first step we use the result of Barrington [6] to reduce checking that an assignment satisfies F to checking that an assignment to variables in the permutation group S_5 satisfies certain equations (over S_5). More precisely, each clause of F gives rise to one equation over S_5. We also provide a way for the verifier to check consistency among distinct occurrences of each literal in F. In the second step we use the randomizing tableaux of Kilian [15] to construct for each equation a 2-prover interactive proof system for an assertion about a product.

The entire proof system is therefore as follows. All parties apply Barrington's result to obtain the set of equations over variables in S_5. The Verifier then randomly chooses either to check consistency or to check that a randomly chosen clause is satisfied. We now describe each of these steps and checks.

2.1 Reduction to Equations over S_5

For reasons of zero-knowledge we first make F a little more "robust" by expressing each variable $y_a \in F$ as the exclusive or of three new sub-variables x_{a1}, x_{a2}, x_{a3}. Note that information about up to three variables in the robust formula gives no information about any variable in F. From now on we simply assume that F is in this robust form.

Following the exposition in [6], a *permutation branching program* of width 5 and depth d is a level graph. Each level is labeled with one of n input variables $x_1, \ldots, x_n$, and contains 5 vertices. Associated with each level ℓ is a pair of permutations $\pi_0^\ell, \pi_1^\ell \in S_5$. Given a setting of the input variables, the level *yields* the permutation π_j^ℓ if the variable associated with level ℓ has value $j \in \{0,1\}$ in the assignment. On input setting $\mathbf{x}$ the branching program yields the product of the permutations yielded by each of the levels. For level ℓ we let g_ℓ denote the variable over S_5 that has value either π_0^ℓ or π_1^ℓ according to the value of the (Boolean) input variable associated with level ℓ.

A permutation branching program B is said to *5-cycle recognize* a set $A \subseteq \{0,1\}^n$ if there exists a five-cycle $\sigma \in S_5 \backslash e$ (called the *output*) such that $B(\mathbf{x}) = \sigma$ if $\mathbf{x} \in A$ and $B(\mathbf{x}) = e$ if $\mathbf{x} \notin A$, where e is the identity permutation.

Theorem 2 *(Barrington [6]): Let A be recognized by a depth d fan-in 2 Boolean circuit. Then A is five-cycle recognized by a permutation branching program of depth 4^d.*

We will apply Barrington's result to a very specific type of circuit: one that checks that the clause

$$(y_{i_1} \oplus y_{i_2} \oplus y_{i_3}) \vee (y_{i_2} \oplus y_{i_2} \oplus y_{i_2}) \vee (y_{i_3} \oplus y_{i_3} \oplus y_{i_3})$$

is satisfied by the input. The clause has at most 9 distinct variables.

We assume that the robust F is a conjunction of clauses of the type just described (that is, F is in a sort of *robust 3-CNF*), so each clause has constant size. For each clause c_i having variables $x_{i1}, x_{i2}, \ldots, x_{i9}$ all three parties create a constant-depth Boolean circuit C_i, which, given an assignment $\mathbf{x_i}$ to the x_{ij}'s, checks that $\mathbf{x_i}$ satisfies c_i. Letting A_i be the set of assignments to $x_{i1}, x_{i2}, \ldots, x_{i9}$ satisfying C_i, the parties then apply Barrington's result to obtain a permutation branching program B_i that five-cycle recognizes A_i. Let $\sigma_i \in S_5$ be the output of B_i. Letting d be the depth of B_i, the construction yields an equation $g_{i1} \ldots g_{id} = \sigma_i$. Here, g_{ij} is associated with the (Boolean) variable that labels level j in B_i, taking on $\pi_0^{i,j}$ or $\pi_1^{i,j}$ according to the value of the associated Boolean variable. Thus, F is satisfiable if and only if (1) for all $1 \leq i \leq m$, the equations $g_{i1} \ldots g_{id} = \sigma_i$ are satisfiable (over the $\pi^{i,j}$'s), and (2) for all ℓ, p, j, q such that the same variable is associated with level ℓ of B_p and level j of B_q, $g_{\ell p} = \pi_0^{p,\ell}$ iff $g_{jq} = \pi_0^{q,j}$ in this satisfying assignment to the g's.

2.2 Checking an Equation

Consider the ith equation $g_{i1} g_{i2} \ldots g_{id} = \sigma_i$. Let us suppress the subscript i for ease of notation, so that we get $g_1 g_2 \ldots g_d = \sigma$. Let $\mathbf{h} = \{h_j | 1 \leq j \leq d, h_j \in \{\pi_0^j, \pi_1^j\}\}$ be an assignment to the g's satisfying the equation. We use a slight modification of the *randomizing tableaux* of Kilian [15] to allow the Provers to convince the verifier of the existence of $\mathbf{h}$.

Let T be the following array with 3 rows and d columns. $T[1,j] = h_j$ for all $1 \leq j \leq d$. Note that $\prod_{1 \leq j \leq d} T[1,j] = \sigma$. Let $r_{1,1}, \ldots, r_{1,d-1}$ be elements

of S_5 chosen independently and uniformly at random. Then $T[2,1] = h_1 r_{1,1}$, $T[2,d] = r_{1,d-1}^{-1} h_d$, and for all $1 < j < d$, $T[2,j] = r_{1,j-1}^{-1} h_j r_{1,j}$. Note that again $\prod_{1 \le j \le d} T[2,j] = \sigma$. Finally, we randomize again, choosing $d-1$ new random elements $r_{2,1} \ldots r_{2,d-1} \in S_5$, and setting $T[3,1] = T[2,1] r_{2,1}$, $T[3,d] = r_{2,d-1}^{-1} T[2,d]$, and for all $1 < j < d$, $T[3,j] = r_{2,j-1}^{-1} T[2,j] r_{2,j}$. Once again $\prod_{1 \le j \le d} T[3,j] = \sigma$. Moreover, neither the second nor the third row of T contains any information about the assignment $\mathbf{h}$.

For any i,j such that $i \in \{1,2\}$ and $j \in \{1, \ldots d\}$, let the i,j *rectangle* be the two entries $T[i,j], T[i+1,j]$. Given the i,j rectangle and the random elements $r_{i+1,j-1}, r_{i+1,j}$ (if $j = 1$ or $j = d$ then only one of these is defined), it is easy to check that $r_{i+1,j-1}^{-1} T[i,j] r_{i+1,j} = T[i+1,j]$. In addition, if T is not a randomizing tableau for $\mathbf{h}, \sigma$ then some rectangle will fail this test [15]. This suggests the following 2-prover interactive proof system.

The Verifier interacts with each prover once. In each interaction it may make the following requests. From P_1 it can request to see *one* of: (1) the third row of the tableau ($T[3,j], 1 \le j \le d$); (2) the i,j rectangle, for some $1 \le i \le 2$ and $1 \le j \le d$.

From P_2 the Verifier can request to see *one* of: (1) an element from the second and third rows of the of the tableau; (2) all the random elements $r_{1,j}, 1 \le j \le d$; (3) all the random elements $r_{2,j}, 1 \le j \le d$; (4) the assignment $\mathbf{x}_j$, where x_j labels one of the levels in the branching program.

The Verifier chooses either to check that the equation is satisfied or that the tableau is correctly constructed. To check that the equation is satisfied, the Verifier requests the third row from P_1 (option (1)) and an element of the top row from P_2 (option (1)). To check that the tableau is correctly constructed, the verifier has three possible options. In all three, it requests an i,j rectangle from P_1.

If $i = 1$: (a) The Verifier can request the assignment to the (Boolean) variable associated with level j. This checks consistency with P_1 and that the $\mathbf{h}_j$'s are chosen from the right sets (the π's). (b) The Verifier can request the randomizers for row 2. This checks that row 2 is formed correctly from row 1. (c) The Verifier can request the element from $T[2,j]$. This checks consistency with P_1.

If $i = 2$: (a) The Verifier can request the randomizers for row 3, checking that row 3 is formed correctly from row 2. (b) The Verifier can request an element from the rectangle, checking consistency with P_1. This completes the description of the protocol.

Intuitively, the most information a cheating verifier can possibly obtain about the bottom row (the assignment $\mathbf{h}$) is the assignment to two of the permutations g_j. Since each of these is associated with only one variable of the *robust* form of the Boolean formula F, and since the values of any two variables in the robust form yield no information about the value of any Boolean variable in the satisfying assignment to the original F, the procedure is truly zero-knowledge. Finally, since the randomizing tableau is for a single clause, it is of constant size. Thus any error in the construction of the tableau is detected with constant probability.

Remark: Checking Consistency

Let x_a be a variable in the robust form of F. Clearly, x_a may appear several times, and it must have the same assignment each time it appears. Let x_a appear in clauses p and q (p and q may be equal). Then for some j, k, x_a is the variable associated with level j of B_p and level k of B_q. Letting $\pi_0^{p,j}$ and $\pi_1^{p,j}$ be the two permutations at level j of B_p, and making analogous definitions for level k of B_q, the verifier must check that $\mathbf{h}_{pj} = \pi_0^{p,j} \Leftrightarrow \mathbf{h}_{qk} = \pi_0^{q,k}$. To check this, the Verifier asks P_1 for the $1, j$ rectangle from the tableau for B_p, and asks P_2 for the assignment $\mathbf{x}_a$ *without* disclosing to P_2 the name of the clause (p or q) that it is examining. This is covered by Case $i = 1(a)$ above.

Remark: Reducing the Number of Shared Random Bits In the description above it was assumed that the random bits used by the provers were completely independent. However, a closer examination reveals that since the verifier never sees more than a constant number of bits, they can be chosen to be c-wise independent for some constant c. Thus, the size of the probability space that generates them can be $O(\log n)$ bits (see e.g. [1]).

2.3 Putting it All Together

Without communicating, the Provers and Verifiers construct the robust form of F and the 5-cycle permutation branching programs for each of the m clauses of the robust form of F. Using their shared random bits, the Provers construct randomizing tableaux for all clauses consistent with a fixed satisfying assignment to the robust form of F. The Verifier randomly chooses a clause and one of the six legal pairs of questions described in the previous subsection, and proceeds accordingly. Note that the Verifier must tell P_1 which clause it has chosen, while it does not tell P_2 the chosen clause when it requests from P_2 the value of an assignment.

We now sketch proofs that our proof system is complete, partially sound and secure.

Theorem 3 (completeness) *If* $\mathbf{x}$, *the assignment known to* P_1 *and* P_2, *satisfies the robust form of* F, *then* V *will always accept.*

Proof. (Sketch) By construction of the randomizing tableaux, a simple case analysis shows that any constraints that V chooses to check will be satisfied.

Theorem 4 (soundness) *There exists a constant* $c > 0$ *such that* V *will reject with probability at least*

$$\frac{c(|S| - \textbf{max-sat}(S))}{|S|},$$

where $\textbf{max-sat}(S)$ *denotes the maximum number of clauses of* S *that may be simultaneously satisfied, and* $|S|$ *denotes the total number of clauses of* S. *This theorem holds regardless of the strategies of the provers,* $\hat{P}_1$ *and* $\hat{P}_2$.

Proof. (Sketch) First, by a standard lemma [7], there exist optimal *deterministic* provers, $\hat{P}_1$ and $\hat{P}_2$, that cause V to accept with the highest possible probability. It suffices to show that even with these provers, V will reject sufficiently often.

$\hat{P}_2$'s responses to queries about $\mathbf{x}$ constitute an assignment. Its responses to queries about rows 2 and 3 of the tableaux define these rows, just as its responses to queries about the randomizers define these objects as well. Let x_i be associated with some level ℓ of B_q, for some clause c_q.

Let c_p be chosen at random. Then with probability at least

$$\frac{c(|S| - \textbf{max-sat}(S))}{|S|},$$

c_p is not satisfied by $\mathbf{x}$. It suffices to show that when this happens V will reject with some constant probability, regardless of what $\hat{P}_1$ does. In this case, $B_p(\mathbf{x}) = e$, so either the product of the elements of the top row of the randomizing tableau for B_p equals e, or the tableau is badly formed. Because the tableau is of constant size the error will be detected with constant probability.

Theorem 5 *The proof system achieves perfect zero-knowledge.*

Proof. (sketch) In order to prove this theorem, we construct a simulator M such that for any satisfiable 3-SAT formula F, any verifier $\hat{V}$ will obtain the same view by interacting with M as by interacting with P_1 and P_2. Recall that in the first step of the interactive proof system, before any communication begins, F is made "robust" by replacing every variable in F with 3 new variables. Let $\mathbf{x}_i$ denote the provers' assignment to x_i in the original formula. Then the provers may choose any random assignment to the sub-variables $x_{i1} \ldots x_{i3}$ so that the exclusive-or of these is $\mathbf{x}_i$.

The verifier makes one of 2 kinds of queries to P_1 and 4 kinds of queries to P_2 for a total of 8 kinds of pairs. The analysis is straightforward; we discuss only the case in which V requests a rectangle from P_1 and an assignment to some x_i from P_2.

Let the (possibly faulty) Verifier request rectangle i, j in the randomizing tableau for B_p from P_1. If $i = 2$ then the rectangle contains two independent randomly chosen elements of S_5, so simulating P_1's response is trivial. If $i = 1$ then since the variable associated with level j of B_p is from the robust form of F, both possible assignments to this variable are equally likely. Thus, either element of $\{\pi_0^{p,j}, \pi_1^{p,j}\}$ is equally likely, so the simulator can choose $T[i, j]$ from this set, and $T[i + 1, j]$ from S_5. Finally, the response from P_2 needs only to be consistent with the response from P_1.

In the final version of the paper we will show how the number of bits that the provers send can be reduced to three - two by one prover and a single bit by the other. Note that this is the best possible, unless $P = NP$, since the existence of a two bit proof system can be translated to a 2-SAT problem.

3 Lower Bound on the Number of Shared Random Bits

In this section we show that the two provers must share $\Omega(\log \log n)$ random bits.

Let r be the number of shared random bits. We make several simplifying assumptions: let the total number of possible queries to each prover be polynomial in n; let the protocol be one round, i.e. the verifier sends the queries to the provers and they respond; let the provers responses be limited to $c \leq 2^r$ possibilities (in this section we do not require c to be constant); let the two provers have no random bits other than the r shared random bits. Some of the above assumptions can be relaxed (see remarks at the end of this section).

We will show that if 3-SAT is recognized by a 2-prover zero-knowledge interactive proof system obeying these constraints and $r \in o(\log \log n)$, then 3-SAT $\in BPP$. The main idea is to first show that a small number of random bits implies that the two provers have only a small number of different strategies for answering the queries. We then show that this implies that on inputs of 3-SAT of any length n, in polynomial time it is possible, using the simulator whose existence is guaranteed by the zero-knowledge property, to reduce the problem to an instance of 3-SAT of size strictly less than n. By repeating this at most n times (a more careful analysis shows that $\log \log n$ times suffice), we can therefore, in polynomial time, reduce the problem to one that can be efficiently solved by brute force.

Fix a satisfiable input formula F of n variables for the remainder of the discussion.

Let $u_1, \ldots, u_m$ ($v_1, \ldots, v_m$) be all the possible queries, over all random choices of the verifier, that the verifier could send to P_1 (P_2). The first step in the reduction is to split the u's (v's) into a (relatively) small number of equivalence classes. We describe the procedure for splitting the u's. The v's are handled similarly.

Intuitively, u_1 and u_2 will be in the same class if P_1 does not distinguish between them. However, for any random string s shared by the two provers, even using the simulator, there is no way to compare the behavior of P_1 on query u_1 with its behavior on query u_2, since each invocation of the simulator queries P_1 exactly once and on different invocations of the simulator the simulated P_1 may have different random strings. We must therefore define the equivalence classes in a slightly more roundabout fashion, so that we can compute them using the simulator.

Let (u, v) be an arbitrary pair of queries to P_1 and P_2, respectively. Let $\text{Answers}((u, v), s)$ denote the pair of responses on this pair of queries when the provers share s. Let $\text{Pairs}(u, v) = \{\text{Answers}((u, v), s) | s \in \{0, 1\}^r\}$. Then

$$u_1 \sim u_2 \Leftrightarrow \forall v(\text{Pairs}(u_1, v) = \text{Pairs}(u_2, v)).$$

Intuitively, although the verifier might distinguish between similar queries, P_1 does not.

At a high level, we will proceed as follows. To reduce the size of the problem we use the simulator to compute the equivalence classes, arguing that there

are not too many of them. The entire strategy of the two provers can then be described by the number of pairs of classes times the number of pairs of responses (c^2, assuming each prover sends one of only c possible answers on each query). But the description of the strategy is just a string, so we have reduced the problem to one of finding a string of at most this length that causes the verifier to accept. We now give more details.

By assumption, the number of u's is at most polynomial in n. We now show that, using the simulator, we can compute $\text{Pairs}(u, v)$ for all pairs of queries u, v, in BPP. For each u we proceed as follows. For each v_i run the simulator many times with a verifier that asks the pair of queries (u, v_i), to obtain the set $\text{Pairs}(u, v_i)$. (It may be that the honest verifier never asks this particular pair of queries. However, some cheating verifier must do so.) Note that as long as the number of shared random bits is at most $O(\log n)$ every element of this set will be discovered with arbitrarily high probability in polynomial time. The sets $\text{Pairs}(u, v)$ are then used to determine the equivalence classes.

Note that for every query u there is a vector of possible replies, each an element in $\{1, \ldots, c\}$, and indexed by the shared random string s. Let this vector of reply be the *color* of the query. There are only c^{2^r} possible colors. Moreover, if two queries have the same color then they are in the same equivalence class (an equivalence class may include queries of different colors). Thus, the number of equivalence classes is at most c^{2^r}. If r is sufficiently small, then we can obtain a representative from each equivalence class on the u's and on the v's. Using the simulator with the real verifier we can obtain, with arbitrarily high probability, for all pairs of representatives (u, v) such that on some execution the verifier actually asks this pair of queries, the set $\text{Pairs}(u, v)$. Call this set a *constraint*. Note that $|\text{Pairs}(u, v)| \leq c^2$.

To reduce the size of the problem, we make the following definitions. Let $u_1, \ldots, u_\ell$ be representatives of the classes of queries to P_1, and let $v_1, \ldots, v_k$ be representatives of the classes of queries to P_2; note that $\ell, k \leq c^{2^r}$. Let S_1 be a function from the representatives u_i to $\{1, \ldots, c\}$, and let S_2 be a function from the representatives v_j to $\{1, \ldots, c\}$. The problem now reduces to finding S_1 and S_2 satisfying the following condition. For all pairs of representatives u_i, v_j such that in some execution of the interactive proof system, V sends a member of the class represented by u_i to P_1 and a member of the class represented by v_j to P_2,

$$(S_1(u_i), S_2(v_j)) \in \text{Pairs}(u_i, v_j).$$

Thus, the problem of proving that $F \in 3 - SAT$ can be reduced to the problem of finding a strategy for the provers that satisfies these constraints. It follows that the question of whether a strategy exists can be defined by a string that is at most the square of the number of classes times the square of the number of possible responses. That is, the length of the description of the constraints that the strategy must satisfy is at most $(c^{2^r})^2 \cdot c^2 = 2^{2^{r+O(1)}}$. Since this question is clearly in NP, it follows from the Cook-Levin Theorem that there exists a polynomial p such that a string x is such a strategy if and only if some (effectively computable) formula F_x of length $p(|x|)$ is satisfiable. Thus,

if $p(2^{2^{r+O(1)}}) < n$ then the original problem of size n can be reduced, in BPP, to a problem of strictly smaller size. This happens when $r = o(\log\log n)$. We therefore have the following theorem.

Theorem 6 *Let L be an NP-complete language recognizable by a perfectly complete perfect zero-knowledge two prover interactive proof system in which the verifier poses a single query to each prover, the reply from each prover is restricted to a single element from a set of size $c \leq 2^r$, and the provers have no random bits other than the shared random bits. Then if the number of shared random bits is $o(\log\log n)$ then $L \in BPP$.*

Note that we have not used the fact that the probability of acceptance in case the formula is not satisfiable is less than $\alpha < 1$ and that the provers are polynomial time machines (with access to a satisfying assignment).

Remarks:

(1) Virtually the same proof shows that the provers must share $\Omega(\log\log n)$ random bits also in *statistical* zero knowledge proofs for NP.

(2) If the provers do not use private random bits, we can assume that the range of possibilities of the provers' replies (denoted by c) is at most of size 2^r. Given that there are only r shared random bits and no private random bits, then on every possible query there are at most 2^r answers that the prover may give. Using the simulator these answers can be enumerated. The protocol can then be changed with the prover giving a pointer of r bits into this list, instead of sending the full answer. The resulting protocol would be only statistical zero knowledge, and would not have perfect completeness. Nevertheless, the proof of the lower bound would still hold with minor modifications.

(3) If the number of possible queries is not polynomial in n then it is still possible, in polynomial time, to find all the equivalence classes that are "likely" to be asked and all the constraints that are likely to influence. The construction proceeds the same way, only we simply ignore "unlikely" queries.

(4) The protocol may contain several rounds instead of one round. The concatenation of a prover's answers plays the role of the prover's answer in the single round case. The only difficulty is in implementing remark 2. However this can be solved by making c no larger than 2^{2^r}, which does not affect the lower bound.

(5) We can allow the provers to have an arbitrary number r of private random bits, provided $\log c + r \in o(\log\log n)$. The main difference is in the definition of the color of a query. In the new definition, each entry in the vector is replaced by a list of possible responses, which vary according to the private random bits of the prover.

(6) Under most assumptions, the lower bound on the number of random bits shared by the provers can be pushed up to $\log\log n - 3$.

(7) While the lower bound shows that $\Omega(\log\log n)$ shared random bits are necessary, the proof relies on the fact that the protocol must be zero-knowledge for *all* verifiers. Indeed, our protocol can be easily modified to use only $O(1)$ shared random bits if zero-knowledge is only required against the honest Verifier.

References

1. N. Alon, J. H. Spencer, **The Probabilistic Method**, John Wiley & Sons, New-York, 1992.
2. S. Arora, C. Lund, R. Motwani, M. Szegedy and M. Sudan, *Proof Verification and the Hardness of Approximations*, Proc. 33^{rd} IEEE Symp. on Foundation of Computer Science, 1992, to appear.
3. S. Arora and M. Safra *Probabilistic Checking of Proofs* Proc. 33^{rd} IEEE Symp. on Foundation of Computer Science, 1992, to appear.
4. L. Babai, L. Fortnow, L. Levin and M. Szegedy *Checking Computations in Polylogarithmic Time* Proc. 23^{rd} ACM Symposium on Theory of Computing, 1991, pp. 21–31.
5. L. Babai, L. Fortnow, C. Lund, *Non-Deterministic Exponential Time has Two-Prover Interactive Protocols*, Proc. 31^{st} IEEE Symp. on Foundation of Computer Science, 1990, pp. 16-25.
6. D. A. Barrington, *Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1*, JCSS (38), 1989, pp. 150-164.
7. M. Ben-or, S. Goldwasser, J. Kilian, A. Wigderson, *Multi Prover Interactive Proofs: How to Remove Intractability*, Proc. 20^{th} ACM Symposium on Theory of Computing, 1988, pp. 113-131.
8. J. Cai, A. Condon, R. Lipton, *On Bounded Round Multi-Prover Interactive Proof Systems*, Proc. of Structure in Complexity, 1990, pp. 45-54.
9. J. Cai, A. Condon, R. Lipton, *Playing Games of Incomplete Information, STACS 1990.*
10. J. Cai, A. Condon, R. Lipton, *PSPACE is Provable by Two Provers in One Round*, Proc. Structure in Complexity, 1991, pp. 110-115.
11. U. Feige, *On the Success Probability of the Two Provers in One Round Proof Systems*, Proc. Structure in Complexity, 1991, pp. 116-123.
12. U. Feige, S. Goldwasser, L. Lovasz, M. Safra, M. Szegedy, "Approximating Clique is Almost NP-Complete", Proc. 32^{nd} IEEE Symp. on Foundation of Computer Science, 1991, pp. 2-12.
13. U. Feige and L. Lovasz, *Two-Provers One Round Proof Systems: Their Power and Their Problems*, Proc. 24^{th} ACM Symposium on Theory of Computing, 1992,
14. L. Fortnow, J. Rompel, M.Sipser, *On the Power of Multi-Prover Interactive Protocols*, Proc. of Structure in Complexity 1988, pp. 156-161. Erratum in Proc. Structure in Complexity, 1990, pp. 318-319.
15. J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, 1990.
16. J. Kilian, *Strong Separation Models of Multi Prover Interactive Proofs*, DIMACS Workshop on Cryptography, October 1990.
17. J. Kilian, *A Note on Efficient Zero-Knowledge Proofs and Arguments*, Proc. 24^{th} ACM Symposium on Theory of Computing, 1992,

18. D. Lapidot, A. Shamir, *A One-Round, Two-Prover, Zero-Knowledge Protocol for NP*, Crypto'91 abstracts.
19. D. Lapidot, A. Shamir, *Fully Parallelized Multi Prover Protocols for NEXP-time* Proc. 32^{nd} IEEE Symp. on Foundation of Computer Science, 1991, pp. 13-18.