Interactive Proof Systems:
Provers that never Fail and Random Selection

(Extended Abstract)

Oded Goldreich
Computer Science Dept.
Technion
Haifa, Israel

Yishay Mansour
IBM Scientific Center
Technion City
Haifa, Israel

Michael Sipser
Mathematics Dept.
MIT
Cambridge, Ma 02139

ABSTRACT

An interactive proof system with *Perfect Completeness* (resp. *Perfect Soundness*) for a language L is an interactive proof (for L) in which for every $x \in L$ (resp. $x \notin L$) the verifier *always* accepts (resp. *always* rejects). Zachos and Fuerer showed that any language having a *bounded* interactive proof has one with perfect completeness. We extend their result and show that any language having a (possibly unbounded) interactive proof system has one with perfect completeness. On the other hand, only languages in *NP* have interactive proofs with perfect soundness.

We present two proofs of the main result. One proof extends Lautemann's proof that BPP is in the polynomial-time hierarchy. The other proof, uses a new protocol for proving approximately lower bounds and "random selection". The problem of *random selection* consists of a verifier selecting at random, with uniform probability distribution, an element from an arbitrary set held by the prover. Previous protocols known for approximate lower bound do not solve the random selection problem. Interestingly, random selection can be implemented by an unbounded Arthur-Merlin game but can not be implemented by a two-iteration game.

1. INTRODUCTION

The two basic notions regarding a proof system are *completeness* and *soundness*. Completeness means that the proof system is powerful enough to generate "proofs" for all the valid statements (in some class). Soundness means that any statement that can be proved is valid (i.e. no "proofs" exist for false statements). Two computational tasks related to a proof system are generating a proof and verifying the validity of a proof. This naturally suggests the notions of a *prover* (a party able of generating proofs) and a *verifier* (a party capable of validating proofs). Typically, the verifier's task is easier than the prover's task. In order to focus on the complexity of the verification task it is convenient to assume that the prover has unlimited power.

For many years *NP* was considered *the formulation* of *"whatever can be efficiently verified"*. This stemmed from the association of deterministic polynomial-time computation with efficient computation. The growing acceptability of probabilistic polynomial-time computations as reflecting efficient computations is the basis of more recent formalizations of "whatever can be efficiently verified". In these formalizations, due to Goldwasser, Micali and Rackoff [GMR] and Babai [B], and shown to be equivalent by Goldwasser and Sipser [GS], the (polynomial time) verifier is allowed to toss coins and arbitrarily interact with the prover, furthermore he can accept or reject based on overwhelming statistical evidence. Ruling by overwhelming statistical evidence means relaxing the completeness and

First author was partially supported by the Fund for Basic Research Administered by the Israeli Academy of Sciences and Humanities. Second author is currently a PhD student at the Laboratory for Computer Science, MIT, Cambridge, MA 02139.

soundness conditions so that any valid statement can be proved with a very high probability while any false statement has only negligible probability to be proved. A bit more formally:

An *interactive proof system* for a language L is a two-party protocol for a *prover* and a *verifier* guaranteeing that, on common input x, when the verifier follows his (polynomial-time) program, the following two conditions hold:

1) *Completeness* (in a probabilistic sense): For every $x \in L$, if the prover follows his program then the verifier accepts x with probability $\geq 1-2^{-|x|}$. In other words, the prover can "convince" the verifier of $x \in L$.
2) *Soundness* (in a probabilistic sense): For every $x \notin L$, no matter what the prover does, the verifier rejects x with probability $\geq 1-2^{-|x|}$. In other words, no prover can "fool" the verifier.

We denote by IP the class of languages for which there exists an interactive proof system. Clearly, $NP \subseteq IP \subseteq PSPACE$. It is believed that the class NP is *strictly* contained in IP. Circumstantial evidence for this can be derived from the fact that, relative to some oracle, interactive proofs are even not contained in the polynomial-time hierarchy, i.e. $\exists A$ s.t. $IP^A - PH^A \neq \emptyset$ (see [AGH]). It is also interesting to note that natural languages as Graph Non-Isomorphism and Matrix Group Non-Membership, which are not known to be in NP, where shown to be in IP (by [GMW] and [B], respectively).

Considering an interactive proof system, it seems that in some sense the prover is "responsible" for the completeness condition, while the verifier is "responsible" for the soundness condition. If this intuition is correct, and the prover has unrestricted power, why should the completeness condition be relaxed? Namely, can one modify the interactive proof such that the prover never fails in demonstrating the validity of true statements, while maintaining soundness. By *perfect completeness* we mean that the prover *never* fails to prove the membership of inputs that are indeed in the language, while *perfect soundness* means that the verifier *never* accepts inputs that are not in the language.

Perfect completeness and perfect soundness are not only theoretically interesting, but are also of practical importance. This is the case, since probabilistic completeness and soundness are defined with respect to ideal (unbiased) coin tosses and may not hold when using pseudo random sequences (even in the sense of [BM, Y]). On the other hand perfect completeness and soundness are independent of the quality of the verifier coin tosses.

Zachos and Fuerer [ZF, Z] have shown that any language having an interactive proof *with a bounded number of interactions* (i.e. the number of interactions is fixed and thus independent of the input) has also such an interactive proof with perfect completeness.

Our main result extends the result of Zachos and Fuerer to interactive proofs with unbounded number of interactions (i.e. the number of interactions may be a function of the input). Namely, we show that ***Interactive Proofs with Perfect Completeness* are as powerful *as Interactive Proofs*.** Now, what about interactive proofs with perfect soundness? Unfortunately, we show that they are only as powerful as NP.

We present two proofs of the main result. Both proofs are in fact transformations that given an interactive proof for a language L yield an interactive proof with perfect completeness for L. In the first proof, we formulate and solve a new protocol problem called random selection (see below). We then use random selection in order to restrict the random choices made by the original Arthur so that Merlin can always convince him in case $x \in L$, while still allowing only a small probability of error in case $x \notin L$. The second proof consists of extending Lautemann's proof [L] that BPP is in the polynomial-time hierarchy$^{(*)}$. We think that the fact that this extension is possible seems strange at first glance, "trivial" in second thought, but is in fact interesting and important.

*) Lautemann's proof that BPP is in the polynomial-time hierarchy simplifies the original proof of Sipser [S].

Let us return to the problem of "random selection". Suppose that the prover has in mind a large set $S \subset \{0,1\}^n$ of size N. The verifier, knowing N, wishes to select at random with uniform probability distribution an element in S. Since the set S is known only to the prover, we can only require that every element selected by the verifier has probability $\frac{1}{N}$, and that there exists a prover such that for every S, all the elements in S can be selected by the verifier. In such a random selection the prover is guaranteed that he is only giving away elements of S, while the verifier is guaranteed that no element is given to him with probability greater then $\frac{1}{N}$ (no matter how the prover plays).

Any protocol for random selection constitutes a protocol for proving approximate lower bound. The converse, however, is false. In fact, all the known protocols for proving approximate lower bounds [B, GS] do not solve the random selection problem. We present a (unbounded iteration) protocol for random selection. Interestingly, we show that there is no two-step protocol which implements random selection. Thus, we demonstrate a meaningful difference between two-step Arthur-Merlin games and unbounded Arthur-Merlin games.

2. MODEL AND DEFINITIONS

Both the Interactive Proof systems (introduced by [GMR]) and the Arthur Merlin games (introduced by [B]) are based on interaction between a powerful *prover* and a probabilistic polynomial time *verifier*. The two parties have a common input x, and they interact in order to allow the verifier to reach a decision about the membership of the input in a specific language. The aim of the prover is to convince the verifier of the membership of the input in the language, while the verifier demands strong evidence to support this claim, but is willing to accept overwhelming statistical evidence. The difference between the two formalisms is that in an Arthur Merlin game the verifier (named Arthur) is limited to send to the prover the outcome of his coins, while in an interactive proof system the verifier can send to the prover a function of his coin tosses and does not necessarily reveal the outcome of the coins. Goldwasser and Sipser [GS] proved the equivalence between the two models; namely that any interactive proof, with $q(n)$ interactions, can be simulated by an Arthur Merlin game with $q(n)+2$ interactions.

We state and prove our main result for the Arthur Merlin games introduced by Babai [B]. Using the result of [GS] our main result applies also to the interactive proof systems of [GMR]. We may assume, without loss of generality, that Merlin is deterministic, since his coin tosses may be approximated by coin tosses which are incorporated into those of Arthur. In the following definition we assume that in all interactions of Arthur and Merlin, on inputs of the same length, the same number of messages are exchanged and that all these messages are of the same length. Clearly, this condition is immaterial and is only placed in order to facilitate the analysis.

Definition 1 (Arthur Merlin games):

An *Arthur Merlin game* is a pair of interactive programs A and M and a function ρ such that:

1) On common input x, exactly $2q(|x|)$ messages of length $m(|x|)$ each are exchanged, where q and m are fixed polynomials.

2) Arthur (A) goes first, and at iteration $1 \le i \le q(|x|)$ chooses at random a string r_i of length $m(|x|)$, with uniform probability distribution.

3) Merlin's reply in the i-th iteration, denoted y_i, is a function of all the previous choices of Arthur and the common input x. More formally, $y_i = M(x, r_1 \cdots r_i)$.

4) For every program M′, a *conversation* between A and M′ on input x is a string $r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}$, where for every $1 \le i \le q(|x|)$ $y_i = M'(x, r_1 \cdots r_i)$. We denote by $CONV_x^{M'}$ the set of all conversations between A and M′ on input x. Note that

$|CONV_x^{M'}| = 2^{q(|x|)m(|x|)}$.

5) The function ρ is a polynomial-time computable mapping of the input x and a conversation $r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}$ to a value $\rho(x, r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)})$. This value is called *the value of the conversation.*

Originally, Arthur Merlin games were introduce to discuss language recognition. In that setting the value of a conversation is either *accept* or *reject* (i.e. ρ is a Boolean predicate). We adopt this convention when discussing language recognition problems. The generalization allows us to discuss a new problem called random selection (see Section 3). For the rest of this section we discuss Arthur Merlin games for language recognition.

Notation: Let **A**, **M′** and ρ be as above. Then $ACC_x^{\rho, M'}$ denotes the set

$$\{r_1 \cdots r_{q(|x|)} \mid \exists y_1 \cdots y_{q(|x|)} \text{ s.t. } r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)} \in CONV_x^{M'} \ \& \ \rho(r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}) = accept\}.$$

Intuitively, $ACC_x^{\rho, M'}$ is the set of all the random choices leading **A** to accept x, when interacting with **M′**. Note that $ACC_x^{\rho, M'}$ depends only on Merlin (**M′**) and the value of the game function (ρ), since we assume that Arthur follows the protocol. The ratio $\frac{|ACC_x^{\rho, M'}|}{|CONV_x^{M'}|}$ is the probability that Arthur accepts x when interacting with **M′**.

Definition 2 (Arthur Merlin proof systems): An *Arthur-Merlin proof system for language L* is an Arthur-Merlin game satisfying the following two conditions:

1) For **M** (as specified in the game), for all $x \in L$, $\frac{|ACC_x^{\rho, M}|}{|CONV_x^{M}|} \geq \frac{2}{3}$. (This condition is hereafter referred to as *probabilistic-completeness.*)

2) For every **M′** and for any $x \notin L$, $\frac{|ACC_x^{\rho, M'}|}{|CONV_x^{M'}|} \leq \frac{1}{3}$. (This condition is hereafter referred to as *probabilistic-soundness.*)

An equivalent definition is obtained by replacing 1/3 by $2^{-p(|x|)}$ and 2/3 by $1-2^{-p(|x|)}$, where $p(\cdot)$ is an arbitrary polynomial satisfying $p(n) > 1$ (for $\forall n > 1$).

Definition 3 (perfect completeness): An Arthur-Merlin proof system with *perfect-completeness* for a language L is an Arthur-Merlin proof system for L satisfying:

$$\forall x \in L \qquad |ACC_x^{\rho, M}| = |CONV_x^{M}|$$

Perfect-completeness, of an Arthur-Merlin proof system, means that an honest Merlin **always** succeeds in convincing Arthur to accept inputs in the language.

Definition 4 (perfect soundness): An Arthur-Merlin proof system with *perfect-soundness* for a language L is an Arthur-Merlin proof system for L satisfying:

$$\forall M' \quad \forall x \notin L \qquad ACC_x^{\rho, M'} = \emptyset$$

Perfect-soundness, of an Arthur-Merlin proof system, means that no matter what Merlin does Arthur **never** accepts an input not in language.

3. RANDOM SELECTION

Definition 5 (Random Selection):
Let $L \subseteq \{0,1\}^*$, $L_n ::= L \cap \{0,1\}^n$, and $l_n = |L_n|$. A *random selection from L* is an Arthur Merlin game (i.e. programs **A** and **M** and a function ρ) satisfying the following:

1) The value of a conversation between **A** and **M**, on input n (in unary) and l_n, is uniformly distributed in L_n. (That is, for every $w \in L_n$, the probability $Prob(\rho(conv) = w) = \frac{1}{l_n}$, where $conv$ is selected with uniform probability distribution from $CONV_{(n, l_n)}^{M}$.)

2) For every **M′**, the value of a conversation between **A** and **M′**, on input n and l, is either a n-bit string or a (special string

called) **error**. Furthermore, for every $w \in \{0,1\}^n$, the probability $Prob(\rho(conv)=w)$ is either 0 or $\frac{1}{l}$, where *conv* is selected with uniform probability distribution from $CONV^{\mathrm{M}'}_{(n,l)}$.

A *random selection protocol* is an Arthur Merlin game (A,M,ρ) such that for every language L, the triplet (A,M(L),ρ) is a random selection from L. (M(L) denotes the execution of M on auxiliary private input L.)

Notation: Let $S^{(\mathrm{M}',l,n)}$ be the set of the values of conversations between A and M′ (on input n and l), excluding the **error** value. (That is, $S^{(\mathrm{M}',l,n)} = \rho(CONV^{\mathrm{M}'}_{(n,l)}) - \{\textbf{error}\}$, where ρ is appropriated extended to sets.) Whenever clear from the context, we use $S^{\mathrm{M}'}$ for $S^{(\mathrm{M}',l,n)}$.

Note that if Arthur has an efficient procedure for testing membership in L then he can check whether Merlin tried to cheat him. In any case Merlin can not enhance the probability of a single string to occur as the value of a conversation. The existence of an efficient procedure for sampling L, yields a trivial solution. However, this is not the case in general, and furthermore is not the case in the applications that we are interested. The reader should note that using a protocol for random selection one can prove approximate lower bounds. (Random selection implies that if $|S^{(\mathrm{M}',2^k,n)}| \leq 2^{k-2}$ then Prob(the value is **error**) ≥3/4.)

3.1. A Protocol for Random Selection

For the design of the protocol, we consider ordered complete binary trees (i.e. each internal node has a *left* and *right* son) of depth k. The following notations are used: (1) $SUCC\ (T,v,d)$ - a function that maps the node v, in the tree T, to its successor using the outgoing edge marked d (e.g. $SUCC\ (T,v,right)$= v's right son). (2) *valued tree* - a tree with integers associated to each node, where $value(v)$ denotes the integer associated with node v. (3) *consistent path* - a path $(v_0 \cdots v_k)$ from the root to a leaf in a valued tree, T, such that for every node v_i if v_{i+1}=$SUCC\ (T,v_i,right)$ then $value(v_i)$ is *smaller* than $value(v_j)$ for all $j \geq i+1$ and if v_{i+1}=$SUCC\ (T,v_i,left)$ then $value(v_i)$ is *greater or equal* to $value(v_j)$ for all $j \geq i+1$. (4) *sorted tree* - a valued tree in which all the paths are consistent.

An overview of the protocol

For sake of simplicity, we present a random selection protocol for the special case that $|L_n|$ is a power of 2. The extension to the general case is easy (see Remark 1 at the end of subsection 3.1).

The protocol proceeds as follows: The common input is a pair $(n,2^k)$. We refer to the program of an honest Merlin which incorporates a set S of size 2^k as an additional input. Merlin creates a complete sorted binary tree of depth k with the leaves containing the elements of the set S. Arthur's aim is to traverse along a random path from the root to a leaf. Arthur creates the path in k steps. At each step Arthur chooses at random a direction to continue the path (*left* or *right*), and Merlin replies with the value of the appropriate node in the tree. This process continues until Arthur has a path of length k. Arthur checks whether the path is consistent. If it is inconsistent then the value of the conversation is **error**, else the value of the conversation equals the value of the last node in the path (i.e. the leaf). This protocol is called *Choose*, and is composed of two interactive programs *Arthur_Choose* $(n,2^k)$ (Arthur's program) and *Merlin_Choose* $(S,n,2^k)$ (Merlin's program). The reader may either think of Merlin as getting the set S as an auxiliary input, or think of Merlin's program as incorporating this set.

Program for Honest Merlin:

Suppose that $|S| = 2^k$ then Merlin acts as follows. Merlin constructs a complete binary sorted tree, $Tree_S$, whose leaves have values in the set S. He sets the value of $node_0$ to the root of $Tree_S$, and initialize the interaction with Arthur by sending him the value of $node_0$ (the root). In step i ($1 \leq i \leq k$), Merlin preforms the following actions:

```
Receive $Direction_i$
$node_i \leftarrow SUCC\ (Tree_S, node_{i-1}, Direction_i)$
Send $value(node_i)$
```

If $|S| \neq 2^k$ then Merlin sends Arthur, at each iteration, an **error** message.

Program for Arthur:

Initially Arthur receives from Merlin the value of the root and saves it in val_0. In step i ($1 \le i \le k$), Arthur preforms the following actions:

```
Choose at random $Direction_i \in \{left, right\}$
Send $Direction_i$
Receive $val_i$
```

The value of a conversation:

The value of a conversation is defined as val_k in case the path is consistent and contains no error messages, and as **error** otherwise. A path is consistent and contains no error messages if and only if $\forall i,j : 0 \le i < j \le k$,
$((Direction_i = left) \Leftrightarrow (val_i \ge val_j)) \ \& \ (val_i \ne \text{error})$
Clearly, the value of a conversation can be computed in polynomial-time.

Remark 1: If $|S|$ (which is input to both A and M) is not a power of two then the programs are modified as follows. Merlin adds to S *dummy* values (larger than any original element in S) to form S' so that $|S'|$ is a power of 2, and runs his program using S'. At each step, Arthur chooses at random a direction, with probability proportional to the number of non-dummy leaves in the different subtrees. Note that with an unbiased binary coin such a probability can only be approximated (with exponentially decreasing error), and thus the definition of Random Selection should be slightly relaxed.

Remark 2: The Random Selection protocol suggested above uses $k = \log_2 |S|$ iterations. By easy modification, we can decrease the number of iterations by a factor of $O(\log k)$, while maintaining the polynomiality of the game.

3.2. Analysis of the Protocol

First we consider the case that Merlin is honest. Let S be the input set of Merlin.

Lemma 1: If $|S| = 2^k$, and Merlin follows his program *Choose_Merlin* $(S, n, 2^k)$, then the value of the conversation is uniformly distributed in S.

Proof: Merlin creates a complete binary sorted tree, and perform the moves, that Arthur requested, on that tree. The value of the conversation is the value of the leaf at the end of the path. All the paths in the tree are consistent, therefore the path that Arthur choose is consistent. All the paths in the tree have the same probability to be chosen by Arthur, hence the value of the conversation is uniformly distributed in $S^{\mathbf{M}}$. □

We show that no matter how Merlin plays, the probability of a single element to be output is either 2^{-k} or zero. For every Merlin, M$'$, we define a binary tree $T^{\mathbf{M}'}$. The tree $T^{\mathbf{M}'}$, describes Merlin strategy, and is not necessarily kept by M$'$. Each node, at level i, corresponds to a possible prefix of i steps in a conversation between M$'$ and A. The left (resp. right) son of a node at the *i-th* level, corresponds to the continuation of the conversation when Arthur's response is *Next=left* (reps. *Next=right*). The value of a node, v, is Merlin's response to Arthur at that iteration and is denoted by $value(v)$. Note that $T^{\mathbf{M}'}$ is a valued tree. The following technical Lemma plays a central role in the analysis.

Lemma 2: In any valued tree, at each level the values of nodes with *consistent* paths from the root, are distinct.

Proof: Assume on the contrary that there is a value that appears in some level on two distinct *consistent* paths. That is, $\exists\, u_1$ and u_2 such that $value(u_1) = value(u_2)$ and both u_1 and u_2 are on consistent paths and at the same level. Consider the least common ancestor of u_1 and u_2, denoted v. Without loss of generality, we assume that u_1 is in the left subtree of v and u_2 is in the right subtree. In order for the paths to be consistent, $value(v) \ge value(u_1)$ and $value(v) < value(u_2)$. Clearly this implies that $value(u_1) < value(u_2)$, contradicting the hypothesis that $value(u_1) = value(u_2)$. □

Lemma 3: For every $v \in S^{\mathbf{M}'}$ the probability that the value of an A-M$'$ conversation is v equals 2^{-k}.

Proof: Recall that $S^{\mathrm{M}'}$ is the set of A-M′ conversation values. The value of a conversation is not **error** if and only if the values of $Direction_i$ and val_i correspond to a consistent path (with no **error** messages) in $T^{\mathrm{M}'}$. Since, $T^{\mathrm{M}'}$ is a valued tree, by Lemma 2, there is a unique leaf (node at level k) with value v and a consistent path from the root. By the one-to-one correspondence between conversations with value in $S^{\mathrm{M}'}$ and consistent paths, the Lemma follows. □

Combining Lemmas 1 and 3, we get

Theorem 4: *Choose* (of Section 3.1) is a random selection protocol. □

3.3. Bounded AM and Random Selection

In the previous subsections we have presented an unbounded Arthur Merlin game for random selection. In this subsection we show that there exists no two-step Arthur-Merlin game for random selection. These results demonstrate a meaningful difference between unbounded Arthur Merlin games and two-step Arthur Merlin games. The reader should note however that this difference does not relate to power of Arthur Melin games as proof systems!

Theorem 5: There exists no two-step Arthur Merlin game for random selection.

Proof's Sketch: Assume to the contrary that (A,M,ρ) is a two-step random selection protocol. Let n be an integer, and C be the collection of all 2^{n-1}-subsets of $\{0,1\}^n$. Consider executions of the game when Merlin's private input is $S \in C$ and the common input is n and $l_n = 2^{n-1}$. For every $S \in C$, define $f(r,S) = \rho(r, \mathrm{M}(S, n, 2^{n-1}; r))$. Clearly, $f(r,S) \in S$. It follows that for every $r \in \{0,1\}^m$ the cardinality of the set $\{f(r,S) : S \in C\}$ is greater than 2^{n-1}. One can show that there exists an $x \in \{0,1\}^n$ such that for most $r \in \{0,1\}^m$ there exists an $S \in C$ such that $f(r,S) = x$. A cheating Merlin, M′, when getting a message r tries to answer with a y such that $\rho(r,y) = x$. By the above, such a y exists for at least half of the r's and thus the value of the game with M′ is x with probability greater than 1/2. The theorem follows. □

We are currently working towards proving the following conjecture

Conjecture: Let t be a fixed integer. Then there exists no t-step Arthur Merlin game for random selection.

3.4. Approximate Lower Bound

Following Babai [B] and Goldwasser and Sipser [GS], we use the following definition.

Definition 6 (Approximate Lower Bound): Let $L \subseteq \{0,1\}^*$, $L_n ::= L \cap \{0,1\}^n$, and $l_n = |L_n|$. An *approximate lower bound for L* is an Arthur Merlin game (i.e. programs A and M and a function ρ), with an oracle to membership in L, satisfying the following:

1) The value of a conversation between A and M, on input n (in unary) and l_n, is *accept* with probability at least 2/3.
2) For every M′ and $l' < \frac{l_n}{2}$, the value of a conversation between A and M′, on input n and l', is *accept* with probability at most 1/3.
3) The value of a conversation can be computed in polynomial-time when having access to an oracle for membership in L.

An *approximate lower bound for L with perfect completeness* is an approximate lower bound for L in which the value of a conversation between A and M, on input n (in unary) and l_n, is always *accept*.

Proposition 6: Let (A, M, ρ) be a random selection from a language L, and $\rho'(conv) = accept$ if and only if $\rho(conv) \in L$. Then (A, M, ρ′) is an approximate lower bound with perfect completeness for L. □

Thus, the protocol *Choose* (together with an oracle for membership in L) is an approximate lower bound with perfect completeness for L.

3.5. Application: Arthur Merlin Proof Systems with Perfect Completeness

In this subsection we sketch the first proof of the main result. An alternative proof is presented in full detail in the next section.

The proof presented here transforms an arbitrary Arthur-Merlin proof into one with perfect completeness, using random selection as a subprotocol. Random selection is used as a substitute for the random choices made by Arthur in the original protocol, so that Arthur's choices are restricted to subsets on which perfect completeness is satisfied.

We denote the original Arthur by $\hat{A}$, the original Merlin by $\hat{M}$, and the original value of the game function by $\hat{\rho}$. Let ε be the error probability, i.e. for $x \in L$ the Prob($\hat{A}$ accepts) $>1-\varepsilon(|x|)$, and for $x \notin L$ the Prob($\hat{A}$ accepts) $<\varepsilon(|x|)$. On input of size n, $q(n)$ iterations are performed, at each iteration Arthur sends a message of length $m(n)$. When clear from the text we use ε, q, m for $\varepsilon(n), q(n), m(n)$, respectively. Without loss of generality, we assume that $\varepsilon<2^{-q-2}$ (see [B], [GS] and [BHZ]).

An overview of the protocol

The main idea is to simulate the original Arthur-Merlin game, in a manner avoiding the conversations in which Merlin fails to convince Arthur of $x \in L$. This is done by Merlin restricting Arthur's choices to subsets favorable to Merlin. Rather then choosing from the set $\{0,1\}^m$, Arthur chooses a m-bit string from a subset of size 2^{m-1}. Arthur's choice is determined by the value of the *Choose* protocol, and is thus guaranteed that any string has exactly probability $2^{-(m-1)}$ (provided that the value is not error). This choice is known to both Arthur and Merlin. Merlin now responses to this choice. At the end Arthur has a conversation, and decides to accept or reject essentially by evaluating $\hat{\rho}$ on the conversation (namely his decision is the same as the one of the original $\hat{A}$'s).

The Computation Tree of an Arthur Merlin Game

Let A and M be programs of an Arthur Merlin game so that on input x they exchange at most $2h$ messages each of length l. The *computation tree* of A and M on input x is a complete valued tree with out-degree 2^l and height h. The nodes are labeled by Merlin's messages and the edges are labeled by Arthur's messages. Each path in the tree correspond to a possible conversation between A and M. The conversation corresponding to the path $v_0 \rightarrow v_1 \cdots \rightarrow v_h$ is $r_1 y_1 \cdots r_h y_h$, where node v_i is marked by y_i and the edge $v_{i-1} \rightarrow v_i$ is marked by r_i. We number the levels of the tree beginning at the root (level zero), and ending at the leaves (level h).

Program for an honest Merlin:

Merlin's program consists of two stages. First, Merlin computes the sets of choices (for Arthur) that are favorable to him. The second stage is a simulation of the original Arthur Merlin game (denoted by $\hat{A}\hat{M}$) when Arthur's choices are restricted to these subsets.

Preprocessing stage

Let T be the computation tree of the original $\hat{A}\hat{M}$ game on input $x \in L$. The aim of Merlin is to find a subtree of T such that there are no leaves that represent conversations that cause Arthur to reject and each internal node has out-degree 2^{m-1}. The subtree is computed in the following manner. First Merlin deletes from T all the leaves that represents conversation that cause Arthur to reject. Next, Merlin performs node pruning level by level, going from the leaves to the root. At each level Merlin prunes all the nodes with out-degree less then 2^{m-1}. The preprocessing is said to have failed, if after the pruning of nodes the final tree is empty, in such a case Merlin enters an **error** mode. If the tree is not empty, Merlin chooses a subtree of T, such that all the internal nodes has out-degree equal 2^{m-1}, and saves it as T. (This is done by a second pruning.)

Simulation stage

Assuming that Merlin did not enter the **error** mode the simulation of the original protocol proceeds as follows. The protocol follows the iterations of the original ($\hat{A}$,$\hat{M}$) game. The current node, is the node in T that represents the prefix of the conversation of $\hat{A}\hat{M}$, up to the present iteration. At iteration i, Merlin restricts Arthur choice to a set of values, denoted by S_i. These values are the labels of the (2^{m-1}) outgoing edges of the current node. After receiving Arthur's random choice r_i, Merlin updates the current node, computes his response, y_i, and sends it to Arthur. Formally, at each iteration i ($1 \leq i \leq q(n)$) Merlin performs:

```
$S_i \leftarrow \{r \mid \exists u \in T : u = SUCC\ (T, current\_node, r)\}$.
$r_i \leftarrow Merlin\_Choose\ (S_i, m, 2^{m-1})$
$current\_node \leftarrow SUCC\ (T, current\_node, r_i)$
$y_i \leftarrow \hat{M}(x, r_1 \cdots r_i)$
        (* $y_i$ is the label of $current\_node$. *)
Send $y_i$
```

In case Merlin enters the simulation stage in the **error** mode he answers all Arthur messages by an error message (i.e. y_i = error for all i).

Arthur's program:

Arthur protocol is identical to the Arthur original protocol, with the difference that instead of choosing the next string locally (at random), it is chosen through the *Choose* protocol. Formally, for each iteration i $(1 \leq i \leq q(n))$ Arthur performs:

```
$r_i \leftarrow Arthur\_Choose(m, 2^{m-1})$
Receive $y_i$
```

The value of a conversation

The value of a conversation is determined by the following polynomial-time predicate
$\hat{\rho}(r_1 y_1 \cdots r_{q(n)} y_{q(n)}) \wedge \forall i \; (r_i \neq \text{error} \; \& \; y_i \neq \text{error})$

Analysis of the protocol

We show that if the input x is in L, then an honest Merlin always convinces Arthur.

Lemma 7: If $x \in L$ then an honest Merlin does not enter the error mode in the preprocessing phase.

Proof's Sketch: Merlin enters the error mode in the preprocessing phase if and only if the pruned computation tree is empty. By backward induction on i, we show that the number of nodes pruned at level i is at most $2^{q-i} \varepsilon 2^{mi}$. Considering level 1 and recalling that $\varepsilon < 2^{-q}$, the Lemma follows. □

Lemma 8: If $x \in L$ then Arthur always accepts.

Proof's Sketch: Use Lemmas 7 and 1. □

We now show that for every input x not in L, no matter what Merlin does, the probability that he convinces Arthur is less then 1/4.

Lemma 9: If $x \notin L$ then Prob(Arthur accepts x) $\leq \frac{1}{4}$.

Proof's Sketch: Consider the execution of the original and simulated games on $x \notin L$. In the original game, the number of accepting conversations is bounded by $\varepsilon 2^{qm}$ no matter how Merlin plays. One can easily see that so is the case in the simulated game. Using Lemma 3, the probability of every accepting conversation in the simulated game is $(2^{-(m-1)})^q$. Hence the probability of Arthur accepting is bounded by $\varepsilon 2^{qm} \cdot 2^{-q(m-1)}$, which equals $\varepsilon 2^q$. Recalling that $\varepsilon \leq 2^{-q-2}$ the Lemma follows. □

Using the equivalence of interactive proofs and Arthur Merlin proofs [GS], and combining Lemmas 8 and 9 we get

Theorem 10: If a language L has an interactive proof system then L has an (Arthur Merlin) interactive proof system with perfect completeness. □

We have showed how to transform an arbitrary Arthur Merlin game into an Arthur Merlin game with perfect-completeness. The transformation has the disadvantage of increasing the number of iteration substantially. The resulting protocol has mq iterations, where q is the number of iterations in the original Arthur Merlin game and m is the size of a messages in that game.

4. ARTHUR MERLIN PROOF SYSTEM WITH PERFECT COMPLETENESS

In this section we transform an Arthur-Merlin proof system to an Arthur-Merlin proof system with perfect completeness. This transformation preserves the number of interactions in the original Arthur-Merlin proof, improving on our original transformation presented in subsection 3.4. The underlying technique for this transformation is taken from Lautemann's proof that BPP is in the polynomial-time hierarchy [L].

Lautemann's technique is commonly presented as a method of expressing a "random" quantifier by a universal and an existential quantifier. Suppose we are dealing with a subset, W, of $\{0,1\}^k$ and that this subset has cardinality either $\geq (1-\varepsilon) \cdot 2^k$ or $\leq \varepsilon \cdot 2^k$. The statement "most $r \in \{0,1\}^k$ are in W" can be substituted by the statement "$\exists s^{(1)}, s^{(2)}, \ldots, s^{(k)} \in \{0,1\}^k$ such that $\forall r \in \{0,1\}^k$ $\exists i$ $(1 \leq i \leq k)$ such that $s^{(i)} \oplus r \in W$", where $s \oplus r$ is the bit-by-bit XOR of the strings s and r. These strings are said to "*cover*" W. The statement "most $r \in \{0,1\}^k$ are not in W" can be substituted by the statement "$\forall s^{(1)}, s^{(2)}, \ldots, s^{(k)} \in \{0,1\}^k$ $\exists r \in \{0,1\}^k$ $\forall i$ $(1 \leq i \leq k)$ $s^{(i)} \oplus r \notin W$".

Zachos showed that the above "simulation" can be used to switch quantifiers in a successive manner (for survey see [Z, Sch]). Zachos and Fuerer [ZF] then used this idea to show that bounded Arthur-Merlin proofs equal bounded Arthur-Merlin proofs with perfect completeness, by expressing the former proofs as a fixed quantifier sequence and applying a "switching lemma" iteratively. Each such iteration is thus a straightforward application of the "simulation technique", and blows-up the size of the Arthur-Merlin game by an unbounded amount. Thus, this idea does not extend to unbounded Arthur-Merlin proofs.

The proof presented in this section uses the simulation technique in a different way. Instead of using it to iteratively switch quantifiers, we extend the simulation technique also to settings in which the witness set W is not predetermined. Particularly, in Arthur-Merlin games the set of random choices leading Arthur to accept when playing with an arbitrary Merlin is not defined.

An overview of the protocol

Without loss of generality, we assume that the error probability in the original Arthur-Merlin game is sufficiently small (i.e. $\varepsilon(|x|) < \frac{1}{3q(|x|)m(|x|)}$). The transformed Arthur-Merlin game will consist of $k = q(|x|)m(|x|)$ original games played concurrently with related coin tosses, and Arthur will accept iff he accepts in one of these games. More specifically, Merlin starts the game by selecting carefully k strings, $s^{(1)},s^{(2)},\ldots,s^{(k)} \in \{0,1\}^k$, and sending them to Arthur. These strings are selected to "cover" $ACC_x^{\rho,\mathrm{M}}$ in the case that x is in the language. Arthur and Merlin now start to play k copies of the original game. In round j, Arthur sends only one m-bit string r_j and his move in the i-th game is defined as the bit-by-bit XOR of r_j and the j-th segment in $s^{(i)}$ (i.e. Arthur's j-th move in the i-th copy is $r_j^{(i)} = r_j \oplus s_j^{(i)}$, where $s_j^{(i)}$ is the j-th m-bit block in $s^{(i)}$). Merlin answers by k strings so that the i-th string equals the answer the original Merlin would have given in the in the i-th copy (i.e. the i-th m-bit block in Merlin's j-th message equals $\mathrm{M}(x, r_1^{(i)} r_2^{(i)} \cdots r_j^{(i)})$, where M is the original Merlin). Clearly, the perfect completeness condition is satisfied. It is less easy to see that probabilistic soundness is satisfied as well. Note that a cheating Merlin may select his answers for one copy of the game depending on his prospects in the other copies, and in particular the structure of $ACC_x^{\rho,\mathrm{M}}$ is irrelevant. Our argument, instead, consists of two claims: 1) the probability of winning the transformed game is bounded by the sum of the probabilities of winning each copy; and 2) the probability of winning a particular copy is bounded by the probability of winning the original game. (Trying to incorporate both claims in one counting argument leads to difficulties which are not encountered in Lautemann's original proof.)

4.1. The Protocol

We denote the original Arthur by $\hat{\mathrm{A}}$, the original Merlin by $\hat{\mathrm{M}}$, and the original value of the game function by $\hat{\rho}$. Let ε be the error probability, i.e. for $x \in L$ the Prob($\hat{\mathrm{A}}$ accepts) $> 1-\varepsilon(|x|)$, and for $x \notin L$ the Prob($\hat{\mathrm{A}}$ accepts) $< \varepsilon(|x|)$. On input of size n, $q(n)$ iterations are performed, at each iteration Arthur sends a message of length $m(n)$. When clear from the text we use ε, q, m for $\varepsilon(n), q(n), m(n)$, respectively. Let $k = qm$. Without loss of generality we assume that $\varepsilon < \frac{1}{3k}$. This can be achieved by performing sufficiently many copies of the original Arthur Merlin game in parallel, and ruling by the majority (see [B], [GS] and [BHZ]).

Program for an honest Merlin:

Merlin's program consists of two stages. First, Merlin computes k "sampling points" that are favorable to him, and sends them to Arthur. The second stage is a simulation of k (related) copies of the original Arthur Merlin game.

Preprocessing stage

Let ACC be the set of random choices leading Arthur to accept in the original $\hat{\mathrm{A}}\hat{\mathrm{M}}$ game on input $x \in L$ (i.e. ACC is a shorthand for $ACC_x^{\hat{\rho},\mathrm{M}}$). Merlin selects k strings $s^{(1)},s^{(2)},\ldots,s^{(k)} \in \{0,1\}^k$ so that for every $r \in \{0,1\}^k$ there exists an i such that $s^{(i)} \oplus r \in ACC$. The preprocessing is said to have *failed*, if no such set of $s^{(i)}$'s exist. If the preprocessing does not fail then Merlin sends the $s^{(i)}$'s to Arthur. For sake of simplicity, we let Merlin send k (arbitrary) strings (of length k-bit each) in case the preprocessing fails.

Simulation stage

Merlin plays concurrently k copies of the original game and computes Arthurs responses by XORing them with segments of the $s^{(i)}$'s. Each $s^{(i)}$ is partitioned into q segments, of m bits each, corresponding to the q iteration of the original game. Namely, $s^{(i)}=s_1^{(i)}s_2^{(i)}\cdots s_q^{(i)}$, where $s_j^{(i)}\in\{0,1\}^m$. Formally, at each iteration j $(1\le j\le q(n))$, Merlin preforms:

```
Receive $r_j$
For $i=1$ to $k$ do begin
        $r_j^{(i)} \leftarrow s_j^{(i)} \oplus r_j$
        $y_j^{(i)} \leftarrow \mathbf{M}(x, r_1^{(i)} \cdots r_j^{(i)})$
End
Send $y_j^{(1)}, y_j^{(2)}, \ldots, y_j^{(k)}$
```

Arthur's program:

Arthur's program is identical to the original program of Arthur. Formally, for each iteration j $(1\le j\le q(n))$ Arthur performs:

```
Choose $r_j$ at random in $\{0,1\}^m$.
Send $r_j$
Receive $y_j^{(1)}, y_j^{(2)}, \ldots, y_j^{(k)}$
```

The value of a conversation

Let $r_j^{(i)}=r_j\oplus s_j^{(i)}$, $y_j=y_j^{(1)}y_j^{(2)}\cdots y_j^{(k)}$, and $\bar{s}=s^{(1)}s^{(2)}\cdots s^{(k)}$. We denote by

$$\rho_i(x,\bar{s}\, r_1 y_1 \cdots r_q y_q)=\hat{\rho}(x, r_1^{(i)}y_1^{(i)}\cdots r_q^{(i)}y_q^{(i)})$$

the value of the i-th game. The predicate ρ_i maps a conversations to 1 if and only if the conversation induced on the i-th copy of the original game is an accepting one.

The value of a conversation is determined by the following polynomial-time predicate

$$\rho(x,\bar{s}\, r_1 y_1 \cdots r_q y_q)=\bigvee_{i=1}^{k}\rho_i(x,\bar{s}\, r_1 y_1 \cdots r_q y_q)$$

4.2. Perfect-Completeness of the protocol

We show that if the input x is in L, then an honest Merlin always convinces Arthur. The argument is almost identical to the one in Lautemann (since ACC is fixed!), and is given here for sake of self-containment.

Lemma 11: If $x\in L$ then the preprocessing does not fail.

Proof: We have to show that if $|ACC|=(1-\varepsilon)\cdot 2^k$ and $\varepsilon\le\frac{1}{3k}$ then there exists a sequence, $\bar{s}=s^{(1)},s^{(2)},\ldots,s^{(k)}$ $(s^{(i)}\in\{0,1\}^k)$, such that for every string $r\in\{0,1\}^k$ at least one of the $r\oplus s^{(i)}$ is in ACC. Furthermore, we will show that the statement holds for most sequences $\bar{s}$. We call a sequence $\bar{s}=s^{(1)},s^{(2)},\ldots,s^{(k)}$ *good* if for every $r\in\{0,1\}^k$ there exists an i $(1\le i\le k)$ such that $r\oplus s^{(i)}\in ACC$. We consider the probability that a randomly selected sequence $\bar{s}$ is not good.

$$\begin{aligned}
&Prob(\bar{s}\text{ is not good})=\\
&\quad=Prob(\exists r\ \forall i\ r\oplus s^{(i)}\notin ACC)\\
&\quad\le\sum_{r\in\{0,1\}^k}Prob(\forall i\ r\oplus s^{(i)}\notin ACC)\\
&\quad=2^k\cdot Prob(\forall i\ s^{(i)}\notin ACC)\\
&\quad=2^k\cdot\varepsilon^k\\
&\quad<(\frac{2}{3k})^k<2^{-k}.
\end{aligned}$$

The Lemma follows. □

Lemma 12: If $x\in L$ then Arthur always accepts.

Proof: By Lemma 11, Merlin can find $s^{(i)}$'s so that (when Merlin follows his program!) any sequence of choices made by Arthur leads to acceptance in at least one of the copies of the original game. The Lemma follows. □

4.3. Probabilistic Soundness of the protocol

We now show that for every input x not in L, no matter what Merlin does, the probability that he convinces Arthur is less then 1/3. We consider the probabilities that Merlin $\mathbf{M}'$ leads Arthur to accept in the i-th copy of the original game. We first bound by ε the probability that $\mathbf{M}'$ leads Arthur to accept in the i-th copy of the original game (see Lemma 13). Hence, the probability that Merlin cheats Arthur is bounded $k\cdot\varepsilon$ (Lemma 14).

Let $\mathbf{M}'$ be any arbitrary program for Merlin. Recall that $ACC_x^{\hat{\rho},\mathbf{M}'}$ denotes the set of random choices leading Arthur ($\hat{\mathrm{A}}$) to accept in the original game (with game value function $\hat{\rho}$). We denote the set of random choices leading Arthur to accept in the i-th game of the transformed game by $ACC_x^{\rho_i,\mathbf{M}'}$. Namely, $r_1r_2\cdots r_q\in ACC_x^{\rho_i,\mathbf{M}'}$ if and only if $\rho_i(x,r_1\mathbf{M}'(x,r_1)\cdots r_q\mathbf{M}'(x,r_1\cdots r_q))=1$. Note that both $ACC_x^{\rho_i,M'}$ and $ACC_x^{\hat{\rho},M'}$ are subsets of $\{0,1\}^k$.

Lemma 13: Suppose that $x \notin L$. Then for every Merlin M′ and for every i $(1 \le i \le k)$

$$|ACC_x^{\rho_i,M'}| \le \varepsilon \cdot 2^k.$$

Proof's Sketch: The idea of the proof is that a Merlin which does well on a particular copy of the original game can be easily transformed into a Merlin which does (at least) as well in the original game. The transformed Merlin (which plays the original game) simulates the actions of the Merlin which plays k games concurrently, using the real game as the i-th copy. A detailed proof follows.

Assume, on the contrary to the statement of the lemma, that there exists an M′ and an i, such that $|ACC_x^{\rho_i,M'}| > \varepsilon \cdot 2^k$. We reach a contradiction by constructing a Merlin M″, which does as well in the original game. First, M″ runs M′ on input x to get the k sample points $s^{(1)},s^{(2)},\ldots,s^{(k)}$ and saves $s^{(i)}$. Let $r_1,r_2,\ldots,r_j$ be the first j messages that M″ has received. To compute the j-th message, M″ computes $r_t' = r_t \oplus s_t^{(i)}$ (for $1 \le t \le j$) and runs M′ on input x and $r_1'r_2' \cdots r_j'$ (i.e. $M''(x,r_1 \cdots r_j)$ is the i-th m-bit block of $M'(x,r_1' \cdots r_j')$). It is easy to see that

$$r \in ACC_x^{\rho_i,M'} \text{ iff } r \oplus s^{(i)} \in ACC_x^{\hat{\rho},M''}.$$

Therefore, $|ACC_x^{\hat{\rho},M''}| = |ACC_x^{\rho_i,M'}| > \varepsilon \cdot 2^k$, which contradicts the hypothesis that the original game has error probability $\le \varepsilon$. The lemma follows. □

Lemma 14: Suppose that $x \notin L$. Then for every Merlin M′ the probability that Arthur accepts is $\le k \cdot \varepsilon$.

Proof: Clearly, for every Merlin M′,

$$|ACC_x^{\rho,M'}| = |\bigcup_{i=1}^{k} ACC_x^{\rho_i,M'}| \le \sum_{i=1}^{k} |ACC_x^{\rho_i,M'}|.$$

Using Lemma 13, the statement follows. □

4.4. Main Result

Using the equivalence of interactive proofs and Arthur Merlin proofs [GS], and combining Lemmas 12 and 14 we get

Theorem 15: *If a language L has an interactive proof system* (with $q(\cdot)$ iterations) *then L has an (Arthur Merlin) interactive proof system with perfect completeness* (and $q(\cdot)+1$ iterations). □

5. INTERACTIVE PROOF SYSTEMS WITH PERFECT SOUNDNESS

In the previous section, we showed that interactive proofs can be modified so that the verifier always accepts valid statements. What happens if we require that the verifier never accepts false statements? In this case we show that the set of languages recognized equals NP.

The difference between interactive proofs and Arthur Merlin games is that in interactive proofs the verifier's i-th message α_i is a function of the input x, his random coin tosses r, and the previous messages of the prover (i.e. $\alpha_i = V(x,r,y_1 \cdots y_{i-1})$. After the last (say q-th) iteration, the verifier decides whether to accept or reject by evaluating the polynomial-time predicate $\rho(x,r,y_1 \cdots y_q) \in \{accept, reject\}$.

Theorem 16: *If a language L has an interactive proof with perfect soundness then $L \in NP$*

Proof: Assume that for a language L, there exists an interactive proof with perfect soundness. Since the verifier is limited to probabilistic polynomial time, then for any input $x \in L$ there is a conversation that convinces him, and is of polynomial length. The NP machine guesses this conversation, checks that it is indeed a legitimate one and that it leads the verifier to accept. Namely, the machine guesses a random tape r and a conversation $\alpha_1 y_1 \cdots \alpha_q y_q$, and checks that $\alpha_i = V(x,r,y_1 \cdots y_{i-1})$ (for every i) and that $\rho(x,r,y_1 \cdots y_q) = accept$. If $x \in L$ then, by the probabilistic completeness condition, there exist (many) accepting conversations. If $x \notin L$ then, by the perfect-soundness condition, there is no such conversation, and any guess of the machine will fail. □

6. CONCLUDING REMARKS

Assuming the existence of secure encryption functions (in the sense of [GM]) and using the results of [GMW], one can easily demonstrate the existence of zero-knowledge interactive proofs with perfect completeness for every language in *IP*. However, it is not clear whether every language having a perfect (resp. almost perfect) zero-knowledge interactive proof (see [F] for definition) has a perfect (resp. almost perfect) zero-knowledge interactive proof with perfect completeness. Weaker statement can nevertheless be proven:

1) Every language having an interactive proof which is almost perfect zero-knowledge *with respect to the specified verifier* has an interactive proof with perfect completeness which is almost perfect zero-knowledge *with respect to the specified verifier* (again see [F] for definition).
2) Every language having an interactive proof which is almost perfect zero-knowledge and remains so under *"parallel composition"* has an almost perfect zero-knowledge proof with perfect completeness.

These statements are proven using the second proof of the main result. (We do not know how to prove them using the first proof of the main result). The key observation in proving both statements is that almost all sequences $\bar{s}$ can serve as sampling points (see proof of Lemma 11), and thus having the prover randomly select and send a *good* $\bar{s}$ does not yield any knowledge.

Goldwasser and Sipser showed that the power of interactive proofs is not decreased when restricting the verifier to use only "public coins" [GS]. We have showed that the power of interactive proofs is not decreased when further restricting the system to have perfect completeness. *How else can interactive proofs be restricted without decreasing their power?*

REFERENCES

[AGH] Aiello, W., S. Goldwasser, and J. Hastad, "On the Power of Interaction", *Proc. 27th FOCS*, 1986, pp. 368-379.

[B] Babai, L., "Trading Group Theory for Randomness", *Proc. 17th STOC*, 1985, pp. 421-429.

[BM] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Jour. on Computing*, Vol. 13, 1984, pp. 850-864.

[BHZ] Boppana, R., J. Hastad, and S. Zachos, "Does Co-NP Have Short Interactive Proofs?", *IPL*, 25, May 1987, pp. 127-132.

[F] Fortnow, L., "The Complexity of Perfect Zero-Knowledge", *Proc. 19th STOC*, 1987, pp. 204-209.

[GMW]Goldreich O., S. Micali and A. Wigderson, "Proofs that yield Nothing But the Validity of the assertion and the a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, 1986, pp. 174-187.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GMR]Goldwasser, S., S. Micali and C. Rackoff, "The knowledge Complexity of Interactive Proof Systems", *Proc. 17th STOC*, 1985, pp. 291-304.

[GS] Goldwasser, S. and M. Sipser, "Private coins versus Public coins", *Proc. 18th STOC*, 1986, pp. 59-68.

[L] Lautemann, C., "BPP and the Polynomial-time Hierarchy", *IPL*, 14, 1983, pp. 215-217.

[Sch] Schoening, U., "Probabilistic Complexity Classes and Lowness", *Proc. 2nd Structure in Complexity Theory Conf.*, IEEE 1987, pp. 2-8.

[S] Sipser, M., "A Complexity Theoretic Approach to Randomness", *Proc. 15th STOC*, 1983, pp. 330-335.

[Z] Zachos, S., "Probabilistic Quantifiers, Adversaries, and Complexity Classes", *Proc. 1st Structure in Complexity Theory Conf.*, LNCS 223, Springer Verlag, 1986, pp. 383-400.

[ZF] Zachos, S., and M. Fuerer, "Probabilistic Quantifiers vs. Distrustful Adversaries", unpublished manuscript, August 1985.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on FOCS*, 1982, pp. 80-91.