# A Note on Time-Space Bounded Interactive Protocols*

Pankaj Rohatgi

TR 91-1198
April 1991

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

This research was supported in part by NSF Research Grant CCR 88-23053.

# A Note on Time-Space Bounded Interactive Protocols *

Pankaj Rohatgi

Department of Computer Science
Cornell University
Ithaca, NY 14853

April 1991

**Abstract**

In this paper, we examine the power of time-space bounded interactive protocols with *private* coins. The class of languages having logspace, polynomial-time bounded private coin protocols is exactly PSPACE. We generalize this result to other time-space bounded protocols. As a consequence we obtain that EXPSPACE is exactly the class of languages having polynomial-space, exponential-time bounded private coin interactive protocols. This coupled with earlier work by Condon, Fortnow and Lund gives us the following characterization of standard complexity classes in terms of time-space bounded interactive protocols.

- DEXP = IPTISP$(exp, poly, public)$
- NEXP = *one-way* IPTISP$(exp, poly)$
- EXPSPACE = IPTISP$(exp, poly, private)$

We also consider time-space bounded *multi-prover* interactive protocols and show that languages in NEXP have 2-prover interactive protocols with logspace, polynomial-time bounded verifiers.

## 1 Introduction

Recent results [LFKN90,Sha90,BFL90] about the computational power of Interactive Protocols (IP) and Multi-Prover Interactive Protocols (MIP) have generated a lot of interest in characterizing the relative powers of different types of interactive protocols. Some of the more recent work has focussed on different types of time-space bounded interactive protocols. Such protocols with *private* coins were first studied by Condon in [Con88], where it was shown that languages in NP have *private* coin protocols with logspace, polynomial-time verifiers.

Using arithmetization techniques, Fortnow and Lund [FL91] recently explored the relationship between time-space bounded *public* coin interactive protocols and time-space bounded alternating Turing machines. They were able to obtain a nice characterization of higher complexity classes in terms of time-space bounded *public* coin interactive protocols. For instance, they showed that DEXP is *exactly* the class of languages having *public* coin interactive protocols with polynomial-space, exponential-time verifiers.

In this paper, we examine *private* coin time-space bounded interactive protocols. Our results imply that the class of languages with private coin polynomial-space, exponential-time interactive

*This research was supported in part by NSF Research Grant CCR 88-23053.

protocols is *exactly* EXPSPACE. This gives a nice characterization of corresponding time and space classes in terms of time-space bounded ***public*** coin and ***private*** coin interactive protocols. These results should be contrasted with Goldwasser and Sipser's result [GS86] which states that private coins do not add to the computational power of polynomial time bounded interactive protocols. The situation is not that nice at lower complexity classes because even though PSPACE is exactly the class of languages with logspace, polynomial-time private coin protocols, the corresponding *public* coin protocols are not known to encompass all of P.

In related work, Condon [Con91] examined one-way time-space bounded interactive protocols in which all communication is from the prover to the verifier. She proved that NP is exactly the class of languages having one-way interactive protocols with logspace, polynomial-time verifiers. These techniques generalize to provide a characterization of NEXP as the class of languages having one-way interactive protocols with polynomial-space, exponential-time verifiers. Thus, for exponential resource bounded classes we now have the nice characterization:

- DEXP = IPTISP$(exp, poly, public)$
- NEXP = *one-way* IPTISP$(exp, poly)$
- EXPSPACE = IPTISP$(exp, poly, private)$

We also explore time-space bounded multi-prover interactive protocols and show that NEXP is exactly the class of languages which have two-prover protocols with logspace, polynomial-time verifiers. Without the provers, such verifiers can only accept languages in P. Thus, we have shown that interaction with the provers helps the verifier, since P $\neq$ NEXP. For ordinary MIPs this is not known because the BPP $\stackrel{?}{=}$ NEXP question has not been resolved.

## 2 Definitions and Notation

We shall use *log* to denote the class of functions $\bigcup_{c>0} c \log n$, *poly* to denote the class of polynomial functions, *ex* to denote the class of functions $\bigcup_{c>0} 2^{cn}$ and *exp* to denote the class of functions $2^{poly}$. We assume that the reader is familiar with the usual complexity classes P, NP, PSPACE, BPP, DEXP = TIME$[2^{poly}]$, NEXP = NTIME$[2^{poly}]$, ESPACE = SPACE$[ex]$ and EXPSPACE = SPACE$[exp]$.

An *Interactive Proof System* consists of two machines, a **Prover** $P$ and a **Verifier** $V$. $P$ and $V$ have access to an input tape and they communicate via a separate communication tape. $P$ is an all powerful Turing machine whereas $V$ is a resource bounded probabilistic machine. The aim of the prover is to convince the verifier that the input is in the language $L$. If the input is indeed in $L$ then $P$ should be able to convince $V$ with high probability. If, on the other hand, the input is not in $L$ then irrespective of $P$'s actions, the verifier $V$ should reject with high probability. More formally, we say that $P$ and $V$ form an *Interactive Protocol* for a language $L$ if:

1. If $x \in L$ then Prob$[P - V$ accepts $x] \geq \frac{2}{3}$
2. If $x \notin L$ then $\forall P^*$, Prob$[P^* - V$ accepts $x] \leq \frac{1}{3}$

Interactive proofs introduced by Goldwasser, Micali and Rackoff [GMR89] require that the verifier be a polynomial time bounded machine. In this paper, we consider protocols in which both time and space are bounded. The power of time-space bounded protocols depends on whether the coin tosses of the verifier are kept private or made public. We shall consider both types of protocols.

We now define various types of time-space bounded interactive protocols. These definitions have been adapted from definitions used in [FL91].

**Definition:** A language $L$ is in IPTISP$(t(n), s(n), \textit{public})$ if there is a *public* coin interactive protocol for $L$, such that, on inputs of size $n$, the verifier uses at most $O(s(n))$ space and $O(t(n))$ time on every computation path with every possible prover.

**Definition:** A language $L$ is in IPTISP$(t(n), s(n), \textit{private})$ if there is a *private* coin interactive protocol for $L$, such that, on inputs of size $n$, the verifier uses at most $O(s(n))$ space and $O(t(n))$ time on every computation path with every possible prover.

We will use the same notation for protocols in which only one resource is bounded. A "$*$" will be used to denote a resource which is not bounded. For example, a language having an $n^2$ time bounded private coin interactive protocol is in the class IPTISP$(n^2, *, \textit{private})$.

One-way time-space bounded interactive protocols form a subclass of time-space bounded interactive protocols. In such protocols all communication is one-way, i.e., from the prover to the verifier. In this model, the prover cannot get to know the verifier's coin tosses and thus these coin tosses are necessarily *private.*

**Definition:** A language $L$ is in *one-way* IPTISP$(t(n), s(n))$ if there is a one-way interactive protocol for $L$, such that, on inputs of size $n$, the verifier uses at most $O(s(n))$ space and $O(t(n))$ time on every computation path with every possible prover.

Multi-Prover Interactive Proofs and the class MIP were introduced by Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] as a generalization of Interactive Proofs and the class IP. Here, instead of just one prover there are polynomially many provers $P_1, P_2, \ldots, P_k$ which *cannot communicate with each other.* Fortnow, Rompel and Sipser [FRS88] showed that polynomially many provers could be replaced by two provers without altering the class MIP. Our definition of MIP will be based on the 2-prover characterization.

**Definition:** Let $P_1$ and $P_2$ be all powerful Turing machines and let $V$ be a probabilistic polynomial time machine. All these machines share the same read-only input tape. $V$ shares communication tapes with $P_1$ and $P_2$ but $P_1$ and $P_2$ do not share any common tape except for the read-only input tape. $V$ forms a multi-prover interactive protocol for a language $L$ if:

1. If $x \in L$ then there exist provers $P_1$ and $P_2$ such that Prob$[P_1, P_2$ and $V$ accept $x] > 1-1/p(n)$, for all polynomials $p$ and large $x$.

2. If $x \notin L$ then for all provers $P_1$ and $P_2$, Prob$[P_1, P_2$ and $V$ accept $x] < 1/p(n)$, for all polynomials $p$ and large $x$.

MIP is the class of languages with multi-prover interactive protocols.

**Definition:** Let $C$ be a class of functions. We say that $C$ is closed under $lin(\cdot)$, if for all $f \in C$ and for all linear functions $l$, the function $l \cdot f \in C$.

Function classes used in complexity theory are usually closed under $lin(\cdot)$. Typical examples include the polynomial functions of various degrees, the polylogarithmic functions of various degrees, and various classes of exponential functions.

## 3 Time-Space bounded Protocols with Private Coins

In this section we examine time-space bounded private coin interactive protocols. First we show that PSPACE is exactly the set of languages having private coin logspace, polynomial-time protocols.

This result also follows from Rompel's result[1] [Rom] that languages in IP have private coin protocols in which the verifiers are logspace, polynomial-time bounded machines.

The proof uses ideas from Condon's proof [Con88] that any $s(n)$ space bounded *public* coin interactive protocol can be simulated by an $O(\log(s(n)))$ space bounded *private* coin interactive protocol (assuming, of course, that $s(n)$ is a reasonable function). Shamir [Sha90] showed that languages in PSPACE have interactive proofs in which the verifier is a logspace, polynomial-time bounded machine ***with two way access to a polynomially long tape containing a random sequence of coin tosses.*** The only communication from the verifier to the prover consists of these coin tosses. This tape has to be hidden from the prover. We show that this tape is not required and the verifier needs to keep only $O(\log n)$ bits of hidden information.

Theorem 2 is a generalization of this result to other space classes. As a corollary, we show that EXPSPACE is exactly the set of languages having private coin interactive protocols in which the verifier is a polynomial-space, exponential-time bounded machine. Fortnow and Lund [FL91] showed that DEXP is exactly the class of languages having polynomial-space, exponential-time *public* coin protocols. Thus, unlike ordinary interactive protocols, the power of time-space bounded protocols depends on whether the coin tosses are made public or kept private. If we consider polynomial-space, exponential-time bounded protocols then *public* coins gave us exactly a deterministic time class and *private* coins give us the corresponding space class. The situation is not that nice at lower complexity classes because we do not have a characterization of the class P in terms of time-space bounded public coin protocols. The best known result [FL91] is that $\text{IPTISP}(poly, log, public) \subseteq \text{P} \subseteq \text{IPTISP}(poly, o(log^2), public)$.

**Theorem 1** $\text{IPTISP}(poly, log, private) = \text{PSPACE}$.

**Proof:** We first show the easy direction, i.e., $\text{IPTISP}(poly, log, private) \subseteq \text{PSPACE}$. We note that $\text{IPTISP}(poly, log, private) \subseteq \text{IPTISP}(poly, *, private)$. It is known that $\text{IPTISP}(poly, *, private) = \text{IPTISP}(poly, *, public) = \text{PSPACE}$ [GS86,Sha90] and the result follows.

We now show that $\text{PSPACE} \subseteq \text{IPTISP}(poly, log, private)$. Let $L$ be any language in PSPACE. By Shamir's result [Sha90], we know that $L$ has a polynomial-time public coin interactive protocol $\mathcal{S}$ involving a prover $P_S$ and a verifier $V_S$ such that:

1. If $x \in L$ then $\text{Prob}[P_S - V_S \text{ accepts } x] = 1$.

2. If $x \notin L$ then $\forall P^*$, $\text{Prob}[P^* - V_S \text{ accepts } x] \leq 2^{-|x|}$.

The idea behind the proof is to simulate the protocol $\mathcal{S}$ using only a logspace verifier with private coins. Unfortunately, a direct application of Condon's proof mentioned earlier gives us protocol in which the verifier is a logspace machine whose expected running time is exponential in the input size. However, by modifying the proof, we can get a verifier which runs in polynomial time.

Intuitively, our protocol requires polynomially many stages. In each stage of our protocol the logspace verifier $V$ asks the prover $P$ to provide an entire run of $\mathcal{S}$. This run consists of a sequence of configurations of the verifier $V_S$ (including the status of the communication tape). A correct run will, of course, depend on the sequence of *public* coin tosses made by $V_S$. In our protocol, $V$ tosses these coins and informs $P$ about what the coin tosses were. Note that the logspace bounded $V$ cannot check that $P$ is providing a valid run of $\mathcal{S}$. It can however check that the start configuration is correct, the end configuration is accepting, the configuration sizes and the running time are bounded and that the state transitions during coin tossing or other operations are locally correct

[1]We give a proof of the $\text{IPTISP}(poly, log, private) = \text{PSPACE}$ result because Rompel's result is cited as a *personal communication* in [Kil88] and we will be using techniques developed in our proof later.

(this requires storing only a small part of the configuration where the heads are). If any of these checks fail then $V$ rejects. Note that $P$ can lie by giving a sequence of configurations which do not follow from one another but nevertheless pass these local tests. Each valid computation history provided by $P$ will look like $c_1\#c_2\#\ldots\#c_{p(n)}$ for some polynomial $p$. Here each $c_i$ is a configuration which is polynomially long and $c_{i+1}$ follows from $c_i$ in one step (either as a normal transition of $V_S$, or as a result of a coin toss made by $V_S$ or as a result of the prover writing one bit on the communication tape of $V_S$ ).

The total size of a valid computation is bounded by a polynomial $q(n)$. If $P$ does not provide a valid computation history then there has to be a configuration $c_{i+1}$ such that $c_{i+1}$ does not follow from $c_i$ in one step.

Now, $V$ can't check that the $c_i$'s follow from one another, but it can randomly select an $i$ and $j$ and check that the contents of the $j^{th}$ tape cell in $c_{i+1}$ follow from the contents of $(j-1)^{th}$, $j^{th}$ and $(j+1)^{th}$ tape-cells in $c_i$ and the head positions in $c_i$. $V$ generates and stores the values of $i$ and $j$ in its private tape so that $P$ does not know where it going to check the configurations. Note that both $i$ and $j$ are bounded by some polynomial and hence require only $O(\log n)$ bits to store. If $P$ provides an incorrect computation history then $V$ will detect this fact with probability at least $1/q(n)$. If $P$ is caught lying then the protocol is terminated and $V$ rejects; otherwise $V$ proceeds with the next stage. After sufficiently many stages $V$ will accept.

Now, if the input $x$ of size $n$ is in the language $L$, then $P_S$ can convince $V_S$ to accept with probability 1. Thus, if $P$ operates like $P_S$ and does not lie about the configurations then $V$ will always proceed to the next stage and finally accept. If on the other hand, $x \notin L$, then in the protocol $\mathcal{S}$, no prover can convince $V_S$ to accept with probability $> 2^{-n}$. Thus, at any stage of our protocol the probability that $P$ can provide a valid computation history is at most $2^{-n}$. Therefore, with very high probability $(1-2^{-n})$, $P$ will have to provide an incorrect computation history and will get caught with probability at least $1/q(n)$ in that case. Thus, if we repeat this protocol $n * q(n)$ times (i.e., have $n * q(n)$ stages) then with high probability $P$ will get caught at least once and $V$ will reject $x$.

**Theorem 2** Let $\mathcal{C}$ be a class of space constructible functions $\geq \Omega(\log n)$ which is closed under $lin(\cdot)$. Then

$$\bigcup_{s(n)\in\mathcal{C}} \text{IPTISP}(2^{s(n)}, s(n), private) = \bigcup_{s(n)\in\mathcal{C}} \text{SPACE}[2^{s(n)}].$$

**Proof:** First we show that

$$\bigcup_{s(n)\in\mathcal{C}} \text{IPTISP}(2^{s(n)}, s(n), private) \subseteq \bigcup_{s(n)\in\mathcal{C}} \text{SPACE}[2^{s(n)}].$$

Let $L \in \text{IPTISP}(2^{s(n)}, s(n), private)$ for some $s(n) \in \mathcal{C}$. Clearly $L \in \text{IPTISP}(2^{c*s(n)}, *, private)$ for some constant $c$. Define the language $L'$ as

$$L' = \{\ x\#^{2^{s(|x|)}} \mid x \in L\ \}$$

Clearly $L'$ has a polynomial time private coin interactive protocol (since $s(n) \geq \Omega(\log n)$). Thus, $L'$ must be in PSPACE. Thus, $L \in \text{SPACE}[p(2^{s(n)})]$ for some polynomial $p$. Therefore, $L \in \text{SPACE}[2^{g(n)}]$ for some $g(n) \in \mathcal{C}$ since $\mathcal{C}$ is closed under $lin(\cdot)$.

Now we show that for any $s(n) \in \mathcal{C}$, $\text{SPACE}[2^{s(n)}] \subseteq \bigcup_{s(n)\in\mathcal{C}} \text{IPTISP}(2^{s(n)}, s(n), private)$. Let $L \in \text{SPACE}[2^{s(n)}]$. Define $L'$ as

$$L' = \{\ x\#^{2^{s(|x|)}} \mid x \in L\ \}$$

$L'$ is in PSPACE. Therefore, by Theorem 1, $L'$ is in IPTISP($poly, log, private$). Thus, $L \in$ IPTISP($p(2^{s(n)}), c * s(n), private$) for some polynomial $p$ and constant $c$. Again, by closure of $\mathcal{C}$ under $lin(\cdot)$ we know that $L \in$ IPTISP($2^{g(n)}, g(n), private$) for some $g(n) \in \mathcal{C}$.

**Corollary 1** ESPACE = IPTISP($ex, linear, private$).

**Corollary 2** EXPSPACE = IPTISP($exp, poly, private$).

## 4 Time-Space bounded Multi-Prover Protocols

The class MIP was introduced by Ben-Or, Goldwasser, Kilian and Wigderson as an extension of single prover protocols. The power of this computational model became apparent only after Babai, Fortnow and Lund [BFL90] used arithmetization techniques to show that MIP = NEXP. Using ideas from Theorem 1, we show that any language in MIP has a multi-prover interactive protocol in which the verifier is a logspace, polynomial-time bounded machine. Feige and Shamir [FS89] have shown that languages in MIP have multi-prover interactive protocols in which the verifier is a finite state machine which runs in polynomial time if the provers are honest and may run for more than polynomial time with very small probability if the provers cheat. We note that our result can be obtained by adding logspace "clocks" to their verifiers but our proof does not require the complicated simulation of work tape that is carried out in their result.

**Theorem 3** Any language in MIP has a 2-prover interactive protocol in which the verifier is a logspace, polynomial-time bounded machine.

**Proof:** We use the Probabilistic Oracle Machine characterization of MIP. It is known that a language $L$ is in MIP iff there is a probabilistic polynomial time oracle machine $M$ for recognizing $L$. The probabilistic oracle machine $M$ accepts $L$ iff

- For every $x \in L$, $\exists O$ such that $M^O$ accepts $x$ with probability 1.
- For every $x \notin L$ and for all oracles $O$, $M^O$ accepts with probability less than $2^{-|x|}$.

Our proof is based on the proof that languages accepted by probabilistic oracle machines have 2-prover interactive protocols [FRS88]. Let us first examine how a language $L$ accepted by a probabilistic oracle machine $M$ can be accepted by a 2-prover interactive protocol. Without loss of generality, we assume that $M$ does not query the same string twice on any of its computation paths. The protocol involves two provers $P_1$ and $P_2$ and a verifier $V$. The protocol runs for polynomially many stages. At each stage, the verifier $V$ simulates one run of the machine $M$ on input $x$. During this simulation, if $M$ makes a query to its oracle $O$ then $V$ asks $P_1$ to provide the answer. $V$ also keeps a list of queries made by $M$ in the current stage along with the answers provided by $P_1$. If the oracle machine $M$ rejects with the given set of answers then $V$ rejects. Otherwise $V$ randomly chooses one of the questions it asked $P_1$ about $O$ during the simulation. $V$ then asks that question to $P_2$. If $P_2$'s reply is different from $P_1$'s earlier response then $V$ rejects. If $V$ has performed the required number of stages then $V$ accepts. Otherwise, $V$ goes to the next stage.

If the input $x$ of size $n$ is in the language $L$, then the provers $P_1$ and $P_2$ can choose an oracle $O$ such that $M^O$ accepts $x$ with probability 1 and answer $V$ accordingly. Thus, the verifier $V$ can be made to accept with probability 1. If, on the other hand, $x \notin L$, then let us analyze the probability that $V$ accepts. At any stage, $s$, $P_2$ is asked only one question. Let $O^s$ be the oracle defined by $P_2$'s response to all the questions that $V$ can ask on any path in stage $s$. Since $x$ is not in $L$, $M$ with oracle $O^s$ will reject on at least $(1-2^{-n})$ fraction of its paths. Thus, on each of these paths, $P_1$ will have to answer differently from $O^s$ at some point in order to prevent outright rejection by $V$. There can be at most $p(n)$ oracle questions on each path of $V$ (here $p$ is a polynomial). Therefore, if at stage $s$, $P_1$'s answers differ from $O^s$, then $V$ will catch the provers lying with probability $\geq 1/p(n)$. Therefore, the probability that the provers are not caught lying at stage $s$ is at most

$$2^{-n} + (1-2^{-n})(1-1/p(n)) \leq 1 - 1/2p(n).$$

Thus, if we have roughly $2np(n)$ stages then with high probability $(1-2^{-n})$, the provers would be caught lying at least once and $V$ will reject.

Some of the salient features of this protocol are that, at every stage, $P_1$ does not know the exact question that $V$ will ask $P_2$, because the choice is made after $P_1$ has answered all its questions. Moreover, at every stage, the lower bound on the probability that cheating provers get caught (i.e., $1/2p(n)$ ), is independent of what happened during previous stages. This result would still hold if we allow the provers to communicate *between* consecutive stages of the protocol and also allowed them to examine $V$'s configuration during this time. These features allow us to give an MIP protocol for $L$ in which the verifier is a logspace, polynomial-time bounded machine with one-bit communication channels.

Our protocol consists of provers $P_1^*$ and $P_2^*$ and a logspace verifier $V^*$. The verifier $V^*$ runs for polynomially many stages. At each stage, $V^*$ asks the prover $P_1^*$ to give a sequence of configurations of $V$ corresponding to *one stage* of the MIP protocol described above. $P_1^*$ provides the sequence of configurations till the point where $V$ asks $P_2$ an oracle question that $P_1$ had previously answered. We assume that this particular question along with $P_1$'s response to it, can easily be recovered from $V$'s configuration. Exactly as in the proof of Theorem 1, $V^*$ tosses the coins for $V$ and checks the start configuration, the size of the configurations and local consistency in the transitions. At the end $V$ asks a question to $P_2$. The last configuration of $V$ given by $P_1^*$ contains the question that $V$ will ask $P_2$ and also the reply expected by $V$. $V^*$ then transfers the question to $P_2^*$ using bit-by-bit communication and rejects if $P_2^*$ answers differently.

Also, as in the proof of Theorem 1, $V^*$ secretly and randomly decides to check consistency of the sequence of configurations provided by $P_1^*$ at some place in the entire sequence. The size of the entire sequence is bounded by a polynomial $q(n)$ and thus $V^*$ needs only logspace to be able to perform this check. If $x \in L$ then clearly $P_1^*$ and $P_2^*$ can convince $V^*$ to accept with probability 1 by not lying. If, $x \notin L$, then at every stage, irrespective of what happened earlier, $V$ rejects with probability at least $1/2p(n)$. Therefore, at every stage, with probability $\geq 1/2p(n)$, $P_1^*$ will have to lie about the sequence of configurations of $V$. In this case $V^*$ will detect the lie with probability $1/q(n)$. Thus, if $x \notin L$ then at every stage $V^*$ will catch the provers cheating with probability at least $1/2p(n)q(n)$. Therefore, if there are more than $n * 2p(n) * q(n)$ stages in the protocol then $V^*$ will reject with high probability.

Note that a cheating prover $P_1^*$ can claim that an arbitrary string was queried by $V$ to $P_2$. The string itself may tell $P_2^*$ how to answer. This does not affect our probability analysis for one round since $P_1^*$ will have to provide a false set of configurations to support this claim and $V^*$ will catch $P_1^*$ with probability $1/q(n)$ in this case. The more serious issue is that this allows transfer of arbitrary information from $P_1^*$ to $P_2^*$. However, as we noted earlier, this does not affect the probability analysis for subsequent rounds although the answers provided by the provers subsequently could depend on what transpired between them in earlier stages.

## Acknowledgements

I am grateful to Juris Hartmanis for his support and guidance. I would also like to thank Suresh Chari, Desh Ranjan and Richard Chang for valuable discussions, comments and suggestions.

## References

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 16–25, 1990.

[BGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove the intractability assumptions. In *ACM Symposium on Theory of Computing*, pages 113–131, 1988.

[Con88] A. Condon. Space bounded probabilistic game automata. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 162–174, June 1988.

[Con91] A. Condon. The complexity of the max word problem and the power of one-way interactive proof systems. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science*, pages 456–465. Springer-Verlag, 1991. *Lecture Notes in Computer Science # 480.*

[FL91] L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science*, pages 263–274. Springer-Verlag, 1991. *Lecture Notes in Computer Science # 480.*

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi prover interactive protocols. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 156–161, June 1988.

[FS89] U. Feige and A. Shamir. Multi-oracle interactive protocols with space bounde verifiers. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 158–164, June 1989.

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

[GS86] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *ACM Symposium on Theory of Computing*, pages 59–68, 1986.

[Kil88] J. Kilian. Zero-knowledge with log-space verifiers. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 25–35, 1988.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[Rom] J. Rompel. personal communication in [Kil88].

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.